Revised Edition

XII

COMPUTER SCIENCE

FOR XIIth BIFOCAL COMPUTER SCIENCE

Computer Software (Paper I)
Computer Hardware (Paper II)

A TEXT BOOK ON THEORY & PRACTICAL

CoMAnd
OPERATING SYSTEMS
DATA STRUCTURES
C++ PROGRAMMING
HTML
MICROPROCESSOR &
INSTRUCTION SET OF 8085
EVOLUTION OF µP
MICRO CONTROLLER
NETWORKING
PRACTICALS
Windows 10

R. D. SUPEKAR
TARAKA GHAMANDE
VAISHALI GHULE
VILAS KARMUDE

A Text Book for Computer Software & Hardware

Computer Science

FOR XII STD. BIFOCAL COMPUTER SCIENCE

COMPUTER SOFTWARE (PAPER I)
COMPUTER HARDWARE (PAPER II)
PRACTICAL (PAPER I & II)

R. D. SUPEKAR
B.Sc. (Physics-Elect.) D.E.R.E., M.C.M. D.I.T
Formerly, HOD Vocational Department Junior College
Lecturer in Electronics and Computer Science,
Teacher in Charge, Vocational Electronics Department,
S.P. College, Pune.
Visiting Faculty Member,
Prin. N.G. Naralkar Institute of Career Development and Research, Pune

MRS. VAISHALI R. GHULE
B.E. (Elect.), Sun Certification, USA
Lecturer in Computer Science, S.P. College, Pune.
Visiting Faculty Member, Prin. N.G. Naralkar Institute
of Career Development and Research, Pune.

MRS. TARAKA GHAMANDE
B.E. (Elect.), DAC
Lecturer in Computer Science,
Sinhgad College of Science
Junior College, Pune.

VILAS KARMUDE
B.Sc. M.C.A A.D.C.A.
Lecturer in Computer Science,
Ramnivas Ruia Jr. College, Matunga, Mumbai.

MAYURESHWAR PRAKASHAN PUNE
Mob: 94 220 180 66

BIFOCAL COMPUTER SCIENCE - XII
A Text Book for Computer Software & Hardware (Theory and Practical)
Paper I & Paper II

Exclusive Rights by : © Mayureshwar Prakashan

First Edition : May 2002 **Second Revised Edition : May 2004**
Third Revised Edition : Jan 2006 **Fourth Revised Edition : July 2007**
Fifth Revised Edition :June 2008 **Sixth Revised Edition : April 2009-2010**
Seventh Edition : May2011-12 **Eighth Edition : Feb 2013-14**
Ninth Edition : May 2015-16 **Tenth Edition : May 2016-17**
Eleventh Edition : April 2018-2019 **Twelfth Edition:May 2019-2020**
Thirteenth Edition : April 2021-2022 **Fourteenth Edition : April 2022-2023**
Fourteenth revised Edition: April 2023-24
Fifteenth Edition : May 2024-25

Published by : Mrs. Varsha R. Supekar

Available At:

- New Students Agencies – Andheri Mumbai [Mobl:902900 2412]
- Vidyarthi sale Agency Girgaon Mumbai [022-23829330]
- Neelkanth Book House – Navi Mumbai [Mobile:96190 60652]
- Pragati Book Centre-Pune [Tel:(020)24486500]
- Natraj Book Point – Pune [Tel:(020)24492106]
- Jagruti Book Agency-Pune [Tel:(020)24486500, 65011663]
- Central Book Stall – Nagpur [Tel:(0712)2555213 Mobile:98230 73969]
- Anupriya Book Agency – Aurangabad [Mob:98502 81146]
- Mahalaxmi Book Stall- Nashik [Mob:942253538]
- Pioneer Book Agency Thane [Mob: 81042 61663]
- Ashish Book Agency – Kolhapur [mob:7798 420 420]
- Dnyansagar- Aurangabad [Tel: (0240)2320885]
- Vikas Book Agency –Pune
- Chinmay Enterprises Pune [Mob: 98501 80089]

Publication : Mayureshwar Prakashan, Pune.
"Rajarshi" S..No. 23/1, Anandvihar colony,Hingne Khurd
Sinhgad Road, Vitthalwadi,Pune – 411051. Mobile:94220 18066

PREFACE

We are glad to publish this new syllabus and the Revised edition of this book. The textbook is prepared by keeping in mind the level of the +2 students offering this subject and the total time they have at their disposal during the year. The main purpose of this book is to provide complete and concise information about the new topics included in the new syllabus. A sincere effort has been made after referring to many prescribed books, manufacturer's data sheets.

This new syllabus published by H.S.C. Board Maharashtra has also mentioned the list of reference books yet practically students have to face several difficulties like unavailability of books, insufficient copies, complex and abstract language and the time for this optional subject for preparation of the notes. This book has been prepared after studying these difficulties sincerely and keeping in mind the level of the science students as well as the aim behind this Computer Science course i.e. more practical aspects than theoretical. The book has been prepared from our teaching experience and suggestions from the teachers teaching this syllabus in different institutions in Pune, Mumbai and Marathwada region. We are very much thankful to them for their important suggestions and valuable help while writing the book.

We hope this will definitely help students and teachers while studying and implementing this new syllabus. A simple and easy language has been used with illustrative diagrams, solved problems and specimen question papers. A great stress has been given to the topics C++ programming, Microprocessor 8085, and Networking with important practical in programming.

Despite the best efforts, it is possible that some unintentional errors might have occurred. We shall acknowledge with gratitude any such errors if pointed out. Any suggestions for the improvement of this book will be gratefully acknowledged and will be definitely eliminated in the next revised edition.

Finally, we express our sincere gratitude to Mr. Adwait Harshe Mrs.Renuka Deshpande, , Prof. Diwate, and our family members for their support in completion of this book.

Rajan Supekar
Vaishali Ghule
Taraka Ghamande
Vilas Karmude

CONTENTS

COMPUTER SOFTWARE (PAPER I)

Sr. No.	Topic	Scope and Limitation	Page No.
1.	**Operating Systems**	• What is Operating System? • Services in OS • Overview of OS:Windows 98,Windows NT, LINUX • Concept related to Information management (Only definition) • File System, Device Drivers, • Terminal I/O • Concepts related to process management (Only definition) • Process, Multiprogramming, • Context Switching, Process States, • Priority, Multitasking, Timesharing. • Concepts Related to Memory Management (Only definition) • A typical map for single user computer, Partitioning, Fixed & Variable Partitioning, Paging, Segmentation, Virtual memory. • GUI : Basic GUI features such as Windows, Task List, Drag, Resize, Close, Minimise, Maximise. • Access and Security aspects of OS • Security Threats, Attacks on Security, Computer Worms, Computer Viruses.	1-31
2.	**Data Structures**	• Introduction to Data Structures, Data Structure. • Arrays – representation in memory, traversing, inserting, deleting, sorting, binary search in an array. Pointers arrays, Records in memory using arrays. • Link List, Representation of link list in memory. • Trees, Binary tree, representing Binary tree in memory.	32-58

3.	**C++ Programming**	• Principle of Object Oriented Programming • Classes and Objects • Constructors and Destructors • Inheritance • Multiple Inheritance • Virtual Functions and Polymorphism • Friends in C++ • Operator Overloading and type conversion • Type conversions • Working with files and streams • File handling in C++	**59-124**
4.	**HTML**	• Introduction to HTML • Why HTML Tags:<HTML>,<HEAD>,<TITLE>,<BODY>,<P>,<BR>,<UL>,<LI>,<PRE>,<MARQ>,<UEE>, • Font Styles • <B>,<I>,<U>,<BIG>,<SMALL>,<SUB>,<SUP>,<FONT>, • Images : <HREF>,<HR>,<IMG>,SRCALT,HEIGHT,WIDTH,ALIGN • List in HTML • Tables <TABLE>,<CAPTION>,<TR>,<TH>,<TD> • HTML scripts (**Note:** Only VB Script using FOR.NEXT. IF..THEN..ELSE, MsgBox, In Box, DIM, SET)	**125-164**
	Paper-I Practicals	C++ Programming, HTML VB	**165-179**

COMPUTER HARDWARE (PAPER II)

Sr. No.	Topic	Scope and Limitation	Page No.
1.	**Introduction to microprocessors and Organisation of 8085**	• Evolution of Microprocessors, • What is microprocessor • Block diagram of Generic microprocessor and study of various Blocks in it.	**181-205**
2.	**Instruction Set and Programming of 8085**	• Addressing Modes in 8085, Programming model of 8085, • Study of Instruction Set • Assembly language programs based on above instructions.	**206-257**
3.	**Introduction to Intel X-86 family**	• Introduction to Advance Microprocessors. • Introduction to X86 Family and study of major attributes of the X86 family processors. • Programming Model of X86 family of microprocessors.	**258-264**
4.	**Introduction to Microcontroller**	• Introduction to Microcontroller, • Study of 8051-Architecture and Programming model, • Overview of other Microcontroller's in the 8051 family, • Applications of Microcontroller.	**265-269**
5.	**Networking Technology**	• Study of Transmission Media-Cable Media-Coaxial, Twisted pair, fibre optic, and their comparison, Introduction to wireless media. • Network topologies-access methods, Topologies- BUS, RING, STAR, Ethernet, Token Ring – Protocols -Internet protocols Introduction to connectivity devices - modem, hubs, repeaters, routers.	**270-293**
	Paper-II Practicals	Assembly language programs.	**294-306**
	Appendix	8085 Instruction Set and Opcode Chart HTML Tags.	**307-310**
	Question Papers I and II	HSC Board Papers for Paper-I and II March2002-March2004,2008	**311-336**

Benefits of XIIth Bifocal Computer Science

Dear students

In XIIth computer science, you will be studying advance C++ programming concepts with basic knowledge of C++ that you have studied in XIth. This year Paper-II Syllabus is based on a new Engineering topic Microprocessor 8085 and assembly language programming.

After completing XII th you will be confident and able to write simple and advance programs in C++, HTML and assembly language programs for 8085 microprocessor. Of course only after studying prescribed syllabus sincerely and making interactions with your theory and practical teachers and also referring this book for various conceptual hints and exercises. These Bifocal courses mostly run by all well known institutions in Maharashtra. These institutions provide well laboratory facilities for both Hardware and Software with Internet facility.

Bifocal courses started in 1978 by Govt. of Maharashtra and revised in 2000 and still they are popular due to their benefits while studying Engineering in any faculty. After completing bifocal computer Science benefits are

- **Exemption for Second language and optional Subject at XI and XII std.**
- **Exposure to Software and Microprocessor Programming**
- **Preparation of Engineering subjects at Junior college level**
- **Preparation of I I T Examination and Preparation of MH-CET Examination**
- **Exemption for Second language and optional Subject at XI and XII std.**
- **Eligible for any Engineering branch (B.E.)**
- **Eligible for Architecture degree course (B.Arch.)**
- **Eligible for Pharmacy (B. Pharm.)**
- **Eligible for Bachelor of Science (B.Sc.) and (BCS)**
- **Eligible for D. Ed. course**
- **Direct Admission to Second year three years Diploma course**
- **Eligible for B.Tech (Agri)**

Since Computer science Bifocal course is of 200 marks. including 100 marks practical, students get much practice and knowledge of this demanding field. Bifocal students will get pleasure in these two valuable and most important years of their academic life and also will get advantage while studying Engineering or IIT subjects.

R.D. SUPEKAR
Mobile: 94220 18066

COMPUTER SOFTWARE

(PAPER-I)

1

OPERATING SYSTEMS

INTRODUCTION

As all know, programs are written for execution. But if we do not know, how to operate computer system then written programs are of no use. We must have the environment within which user as well as other programs can do useful work. How this environment is provided by operating system is discussed in this chapter.

1.1 IDEA OF AN OPERATING SYSTEM

An operating system is a program, which acts as an interface between user of a computer and computer hardware. It provides an environment to run other programs also. It makes system convenient to use. It also performs basic tasks like input and output management i.e. it helps in using hardware in efficient manner.

Any computer system has following layer structure.

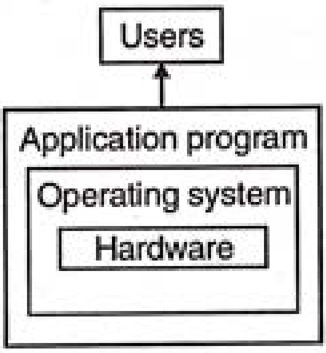


Fig. (1.1)

Computer hardware is at innermost level. Operating system is at outer level to that of computer hardware. Application programs are at outer level of the operating system. While users interact with the computer hardware through application program and operating system.

Services in O/S

Operating system (O/s) provides certain services to the programs and to the users. There are following three major types of services:

1) Information Management
2) Process Management
3) Memory Management

Information Management

It refers to a set of services used for storing, retrieving, modifying or removing the information on various devices. It manages the organization of information in terms of directories and files, allocating, deallocating the sectors to various files ensuring right people have access to information and driving various devices.

Some of the system calls in this category are as follows:

1) Create a file
2) Create a directory
3) Open a file (for, read, write or both).
4) Close a file
5) Read data from file to buffer
6) Move the file pointer
7) Read and return file's status
8) Create a link
9) Change working directory
10) Write data from buffer to file

Process Management

If operating system supports multiple users then services under this are very important. In this regard operating system has to keep track of all the completing processes (running programs), schedule them, dispatch them one after another. But user should feel that he has the full control of the CPU.

Some of the systems calls in this category are as follows;

1) Create a child process identical to the parent
2) Terminate a process
3) Wait for a child process to terminate
4) Change the priority of process
5) Block the process
6) Ready the process
7) Dispatch a process
8) Suspend a process
9) Resume a process
10) Delay a process
11) Fork a process

Memory Management

The services provided under memory management are directed to keeping track of memory and allocating, deallocating it to various processes. The operating system keeps a list of free memory locations. Before a program is loaded in memory from the disk, this module consults this free list, allocates the memory to the process, depending upon program size and updates the list of free memory.

Some of the systems calls in this category are as follows,

1) Allocate a chunk of memory to process
2) Free a chunk of memory from a process

Overview of O/S

Operating system first developed by Microsoft was DOS. But it was a single user operating system. After DOS a complete new system with graphical user interface (GUI) was developed. It was windows 3.1. Later on Windows 95 an upgraded version of windows was launched. After that Windows 98 a new version of windows was launched. It is multitasking and again GUI based operating system. Some of the main features of windows 98 are listed below;

1) Faster operating system

It includes tools that help your computer run faster than Windows 95 without adding new hardware. It optimizes your computer's efficiency by a group of some special programs that are used together some of the programs are like

1) Maintenance wizard, 2) Drive converter, 3) Disk defragmenter.

2) Easy to use

It makes your computer easier to use with new and advanced features like web integration, multiple display support, power management, and universal serial bus.

Web integration

Windows 98 explorer and Internet explorer integrate local and web based resources in a single view. Auto complete automatically completes previously visual addresses as you type them. It supports for all major Internet standards ActiveX, Java.

Multiple display support

It makes possible to use several monitors simultaneously to increase size of your desktop.

Power management

On-Now makes your computer more responsive by improving startup time. Using power management techniques On-Now can start your computer in just a few seconds and restore all your programs where you left them.

Universal serial bus

It makes your computer easier to use with advanced plug and play capabilities. Using a new universally standard connector you can add devices to your computer easily without having to restart.

3) More Entertaining and Fun

It makes your computer more entertaining by introducing new features such as enhanced television, video playback and support for new hardware.

4) Help

It is very easier to use and find answers to questions quickly. You can also get up todate technical support from World Wide Web (WWW).

1.2 WINDOWS NT

Windows NT is multi-user, multitasking and multithreading operating system. Multitasking is preemptive i.e. each task is given time slice when that time slice is over the next one is started.

Windows NT features include virtual memory management, symmetric multiprocessing. Symmetric multiprocessing allows Windows NT to schedule various tasks on any CPU. Generally tasks are scheduled on any available processor. When tasks are broken into independent threads, a large amount of parallelism is achieved.

Windows NT can interact with all existing networking systems like Novell's Netware, Sun micro system's NFS, etc. It is also portable operating system and will be available on other hardware platforms like Intel's X-86 family.

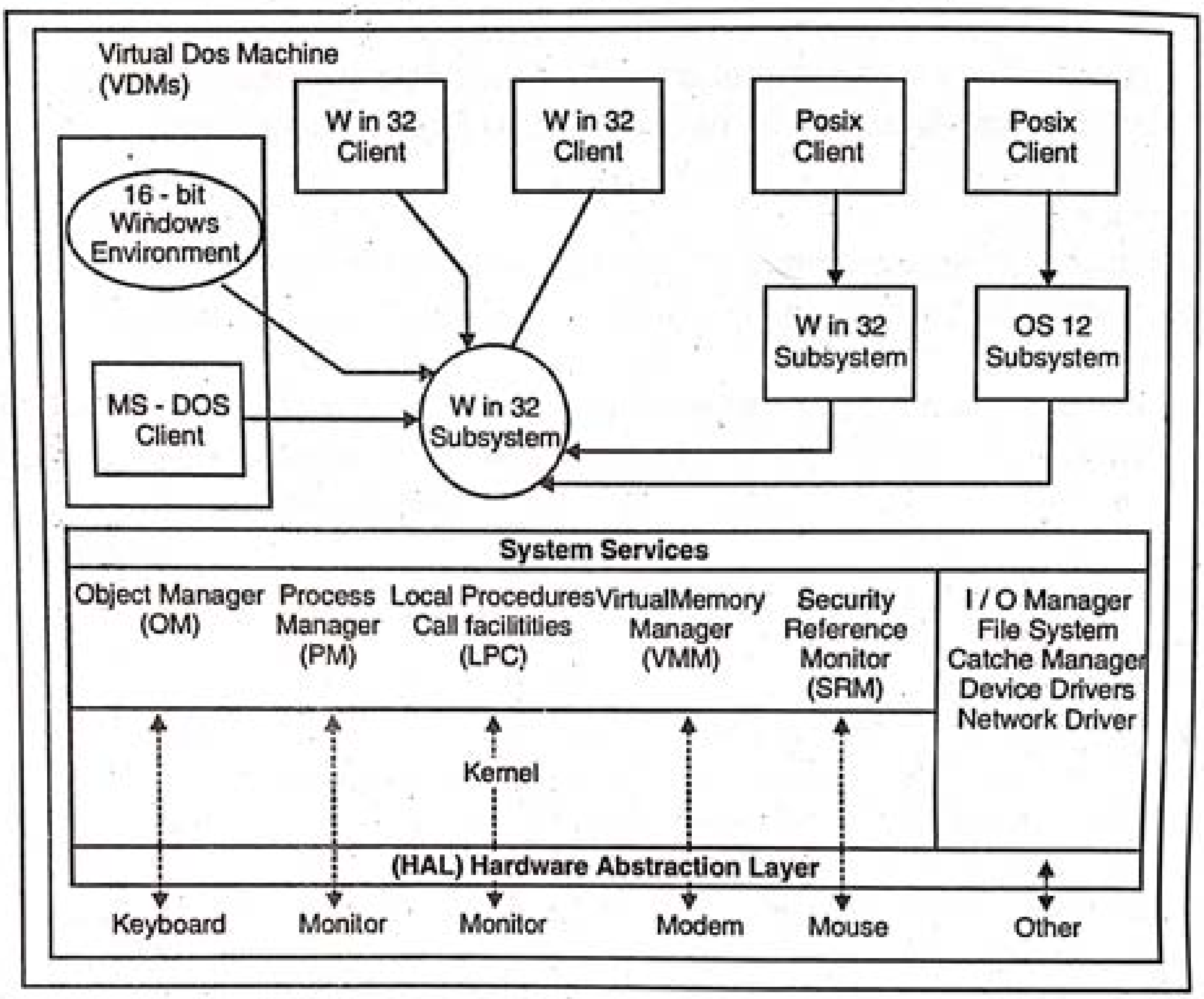

Virtual Dos Machine
(VDMs)
W in 32 Client
W in 32 Client
Posix Client
Posix Client
16 - bit Windows Environment
W in 32 Subsystem
OS 12 Subsystem
MS - DOS Client
W in 32 Subsystem
System Services
Object Manager (OM)
Process Manager (PM)
Local Procedures Call facilitities (LPC)
VirtualMemory Manager (VMM)
Security Reference Monitor (SRM)
I / O Manager
File System
Catche Manager
Device Drivers
Network Driver
Kernel
(HAL) Hardware Abstraction Layer
Keyboard
Monitor
Monitor
Modem
Mouse
Other

Fig. (1.2) Architecture of Windows NT

Windows NT has a new high performance file system called New Technology File System (NTFS) implements fault tolerance, security and has support for large files.

Windows NT has a modular architecture divided into different subsystems and layers. At lowest layer is hardware abstraction layer. It provides all hardware specific functions. On top of that is the Windows NT kernel. It provides basic operating system services. On top of kernel there are six modules, which are listed below with their functions.

1) Object manager- it creates manages and deletes objects, which are processes.
2) Process manager- it creates, terminates, suspends and resumes processes, tasks, and threads.
3) Virtual memory manager- it manages the memory for processes. It also allocates and frees memory.
4) Security reference monitor- It enforces security policy and keeps track of file access rights based on ownership and permission for the user.

5) Local procedures call facility- it is used to pass messages between client systems and subsystems on one system.
6) I/O subsystem- it handles device and passes data to and receives data from the devices of all subsystems.

1.3 LINUX

Linux is an operating system that was initially created as a hobby by a young student, Linus Torvalds, at the University of Helsinki in Finland. Linus had an interest in Minix, a small UNIX system, and decided to develop a system that exceeded the Minix standards. He began his work in 1991 when he released version 0.02 and worked steadily until 1994 when version 1.0 of the Linux Kernel was released. The current full-featured version is 2.4 (released January 2001) and development continues.

Linux is developed under the GNU (General Public License) and its source code is freely available to everyone. This however, doesn't mean that Linux and its assorted distributions are free and developers may charge money for it as long as the source code remains available. Linux may be used for a wide variety of purposes including networking, software development, and as an end-user platform. Linux is often considered an excellent, low-cost alternative to other more expensive operating systems.

Due to the very nature of Linux's functionality and availability, it has become quite popular worldwide and a vast number of software programmers have taken Linux's source code and adapted it to meet their individual needs. At this time, there are dozens of ongoing projects for porting Linux to various hardware configurations and purposes.

Linux has an official mascot, the Linux Penguin, which was selected by Linus Torvalds to represent the image he associates with the operating system he created. Linux is a reliable, secure operating system. In addition to being cost-effective, it is constantly being updated and refined with the latest technologies. As Linux gains greater acceptance throughout the computing industry, more and more companies are supporting Linux via both application and hardware compatibility.

Features of Linux

1) Linux is Network-friendly

Since a team of programmers developed Linux over the Internet, its networking features were given high priority. Linux is capable of acting as client and/or server to any of the popular operating systems in use today, and is quite capable of being used to run Internet Service Providers.

Linux supports most of the major protocols, and quite a few of the minor ones. Support for Internet, Novell, Windows, and Appletalk networking have been part of the Linux kernel for some time now. With support for Simple Network

Management Protocol and other services (such as Domain Name Service), Linux is also well suited to serving large networks.

2) Linux is a Multi-user

Linux is an implementation of the UNIX design philosophy, which means that it is a multi-user system from the word "go." This has numerous advantages, even for a system where only one or two people will be using it. Security, which is necessary for protection of sensitive information, is built into Linux at selectable levels. More importantly, the system is designed to multi-task. Whether one user is running several programs or several users are running one program, Linux is capable of managing the traffic.

3) Linux is Open

Linux is open. That means that for the entire base system, which includes the kernel, the GNU tools, and all the basic utilities, we as programmers and users have access to the source code as well as the right to modify it.

4) Linux is "Free"

Linux consumer is free to modify the system and do anything he or she wishes with it.

Information Management consists of two main modules,

1) File system 2) Device driver

1.4 FILE SYSTEM

The file system allows the user to define files and directories and allocate and deallocate the disk space to each file. It uses various data structures to achieve this.

For information storage, O/S uses files. That information is stored on files. Files are mapped by the operating system on to physical devices.

A file is a collection of related information. It can contain programs and data. Data may be numeric, alphabetic or alphanumeric. File is a sequence of bits, bytes, lines or records. Each file is having name and extension. (It is dependent on file type)

Block

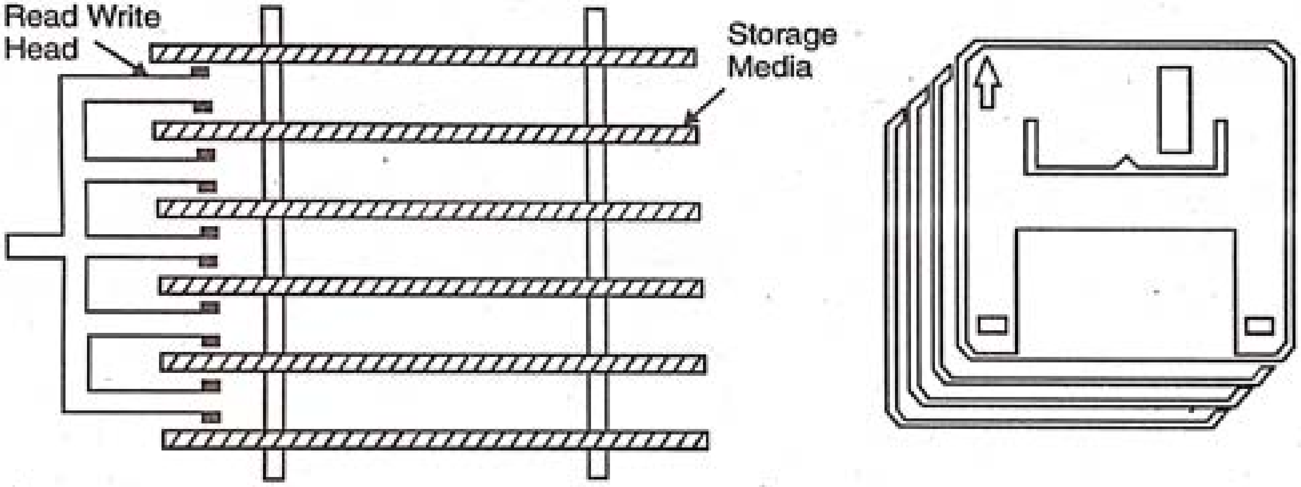
It is a unit of data, which the operating system defines for the sake of convenience. Normally operating system also keeps all its data structures in terms of blocks. Operating system would view the disk as comprising of no. of blocks. From hardware point of view a disk consists of no. of sectors, but from the point of view of the operating system, it consists of no. of blocks where each block is having one or more sectors. Due to this operating system has to translate a block number into physical sector numbers.

Disk

It is main secondary device. Various types of secondary devices are there like floppy disk and hard disk. Hard disk can be considered as being made up multiple floppy disks put one above the other.

Floppy Disk

Floppy disk is made up of round piece of plastic material coated with a magnetized recording material. The surface of a floppy disk is made of concentric circles called tracks. Disk can be considered to be consisting of several surfaces, each of which consisting of tracks. The tracks are normally numbered from as the outermost track with the number increase inwards. Each track is divided into a no. of sectors of equal size. Sector capacities may vary; but typically a sector can store up to 512 bytes.



Hard Disk Mechanism Fig. (1.3) Floppy Disks

Double-sided floppy have two sides on which data can be recorded, so given sector is specified by a components of address, surface no, track no, and sector no.

Hard Disk

Hard disk surface is also made of concentric circles called tracks. Each track is divided into sectors of equal size. The disk drive for hard disk and its read/write mechanism is shown in fig. (1.3). Read/ write head for each surface connected to arm. Arm can move in or out to position itself on any track while disk rotates at a constant speed e.g.

1) Read a block
2) It is translated into a sector address surface=1, track=10, sector=5. This is called target address.
3) Read/write heads current position's address is current address, i.e. Surface =0, track =7,sector =7.
4) Move the arm in or out position on correct track. In this e.g. move arm inwards from track 7 to track10. The time taken to move to correct track is called seek time.

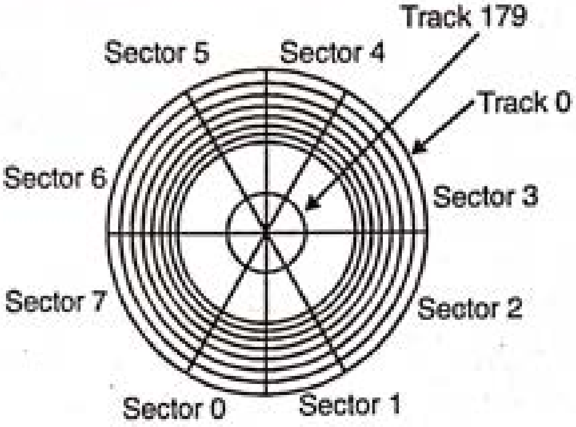

Track 179
Sector 5
Sector 4
Track 0
Sector 6
Sector 3
Sector 7
Sector 2
Sector 0
Sector 1

Fig. (1.4a) Tracks and Sectors

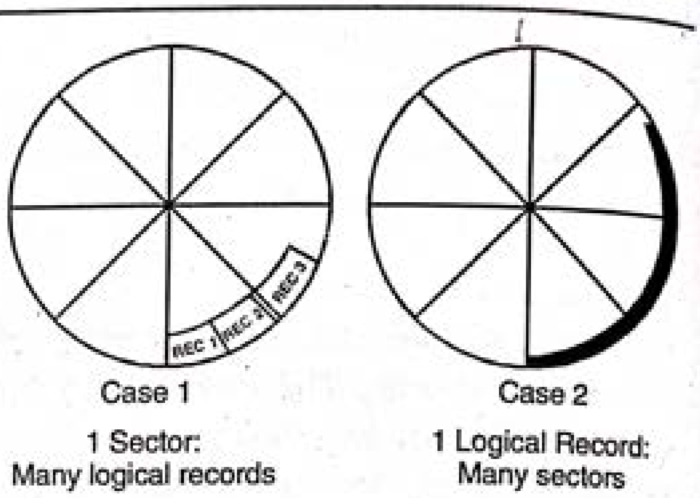

REC 1
REC 2
REC 3
Case 1
1 Sector:
Many logical records
Case 2
1 Logical Record:
Many sectors

Fig. (1.4b) Logical records and sectors

5) Wait until desired sector comes under r/w head as disk rotates. Time to reach to particular sector on track is called rotational delay or latency.

6) Activate R/W head for appropriate surface and read data. The time taken for this operation is called transmission time.

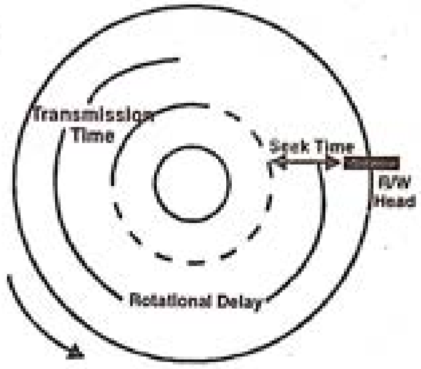

Transmission Time
Seek Time
R/W Head
Rotational Delay

Fig. (1.5) Stages Of I/O Operations

File

A collection of related information is termed as file. For each file created by the operating system maintains a directory entry also called volume table of contents (VTOC) as shown.

File Name
File Size
No. Of Processes Having This File Open
File Record Length
File Owner
File Address Of First Block
File Extension
File Usage Count
File Type (Binary/ASCII)
File Organization
File Dates (Creations, Last Usage)

VTOC Structure

FILE OPERATIONS

General file operations include create, read, write, rewind & delete a file.

1) Create a file

For creating a file, first whether sufficient space is available for that file is checked. If it is available, entry for new file must be made in directory.

2) Write to a file

For writing to file, there is command in which name of the file is given. Then operating system search for that file in directory entry and write to it.

3) Reading a file

For reading also there is system call in which file name is specified. Then operating system searches for required file in directory entry and read it.

4) Rewind a file

The directory is searched for appropriate entry and file is reset to the beginning of file.

5) Delete a file

To delete a file, again directory entry is searched & if that file is found it releases the memory space and that directory entry now become invalid.

1.5 ALLOCATION METHODS

For the allocation of disk space to various files, two major philosophies are there, 1) Contiguous 2) noncontiguous.

In this scheme user estimate maximum file size that the file will grow to, considering the future expansion and then requests the operating system to allocate those blocks at the time of creation of file.

Operating system normally uses block allocation list to manage the disk space. The list gives the account of all blocks on the disk. It also gives you whether the block is free or allocated. It is more useful to maintain two different tables:
1) allocated blocks 2) free blocks.

Whenever a new file is created depending upon the size of file operating system can allocate continuous area. If there are multiple free blocks then which should be chosen? For that following methods are used?

1) First fit: If suppose file requires 7 blocks; then OS goes through block list and allocate the entry which blocks equal or more than 7.
2) Best fit: For this method free blocks list is to be sorted by no of free blocks. It would choose that entry which is the smallest amongst all the entries, which are equal to or bigger than required one.
3) Worst Fit: It allocates free block that is equal to or larger than our requirement. The main disadvantage of this method is space wastage & inflexibility. Until file grows to maximum size, many blocks allocated to the file remain unutilized. So

lot of space is wasted. Again file grows more than predicted maximum it creates a real problem and next blocks have been allocated to another file.

Noncontiguous Allocation:

Considering the problems in contiguous allocation, this method is evolved. In this maximum size of file does not have to be predicted at the beginning. The file can grows with time as per the needs. This reduces wastage of space and operating system automatically allocates additional blocks as per requirement without aborting program. There are two methods for implementing this allocation.

(1) Chained allocation (e.g.MS-DOS)
(2) Indexed allocation (e.g. Unix)

Chained allocation (e.g. MS-DOS)

This is noncontiguous allocation. So next block is traversed using pointer field, which gives address of next block in same file.

In DOS, these pointers are kept externally in file allocation table.

In MS-DOS file allocation table is used as shown in fig. (1.6). In this scheme the block can have 512 or 1024 bytes as shown in fig. there are three files in our system. File A has been allocated blocks 5,7,3,6,10. File B has been allocated blocks 4,8,11. File C has been allocated blocks 9,2 and 12. The file directory entries for these three files have been shown in fig.

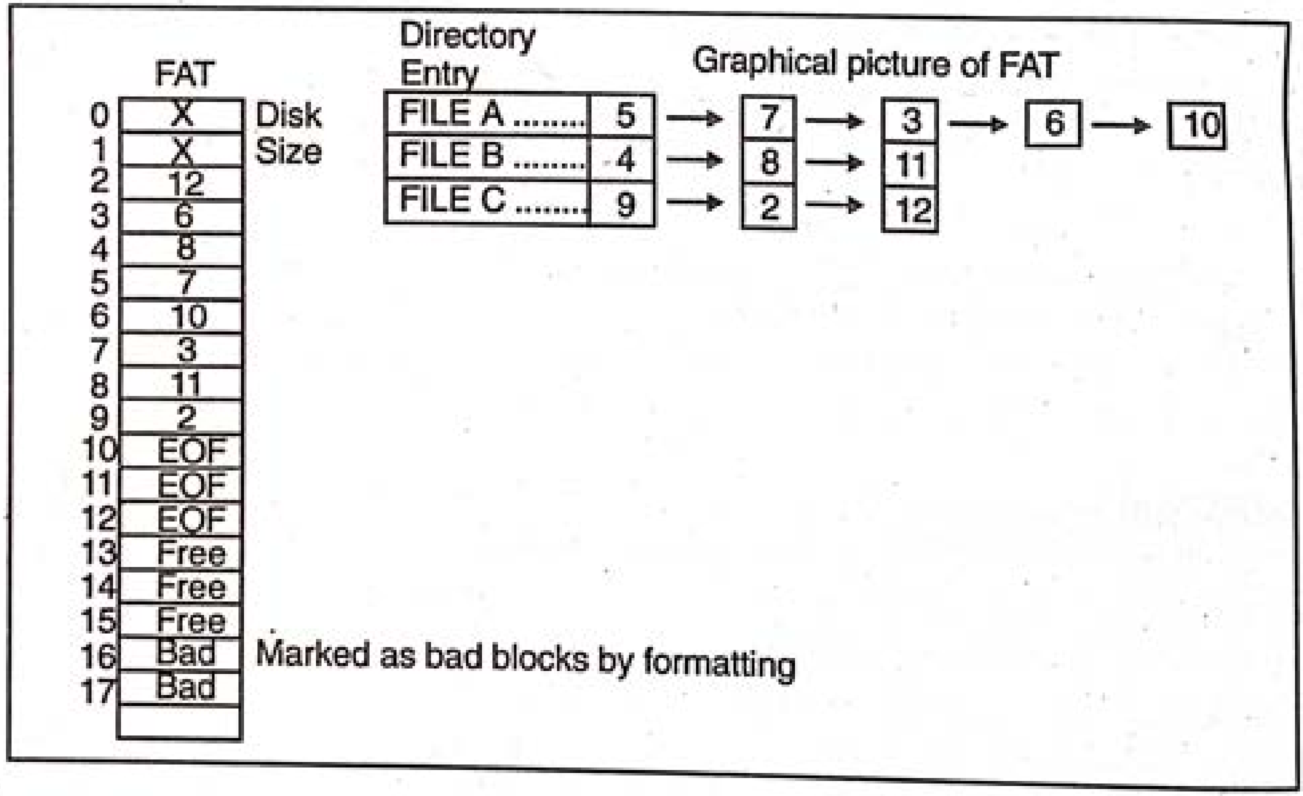
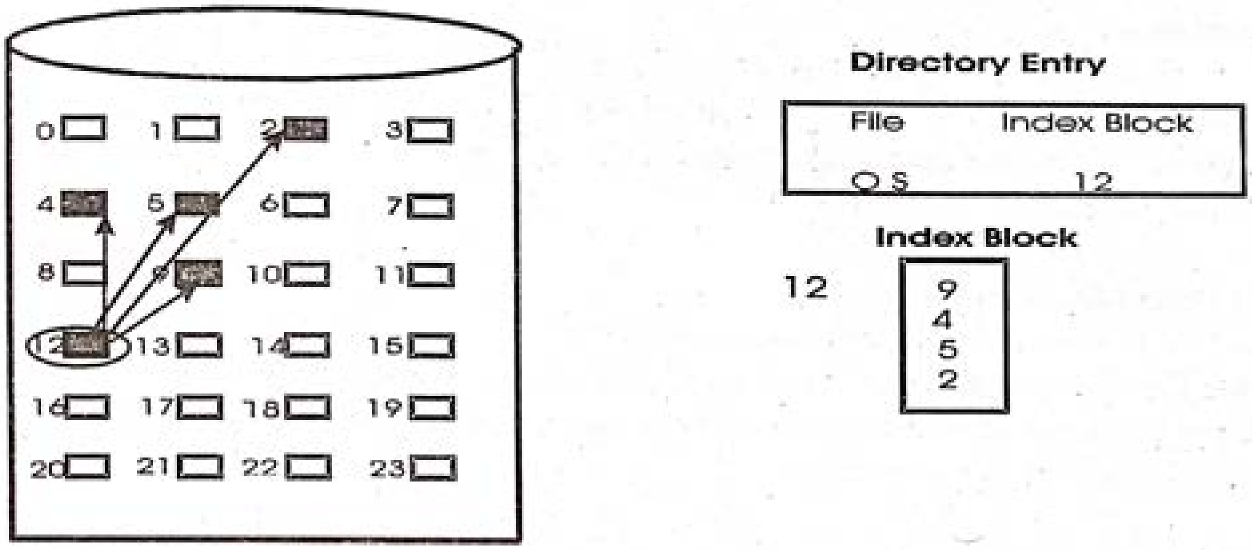

FAT
Disk
Size
Directory Entry
Graphical picture of FAT
FILE A 5
FILE B 4
FILE C 9
7 → 3 → 6 → 10
8 → 11
2 → 12
0 X
1 X
2 12
3 6
4 8
5 7
6 10
7 3
8 11
9 2
10 EOF
11 EOF
12 EOF
13 Free
14 Free
15 Free
16 Bad
17 Bad
Marked as bad blocks by formatting

Fig.(1.6) MS-DOS Chained Allocation

Indexed Allocation

Index of blocks is maintained. It is externalized list of pointers. Then the file directory entry should point to the index.

Index is maintained in various ways. For example for a file, list of pointers is as shown below



As shown in figure

1) Index block is 12. It is specified in directory entry.
2) In Index block, all block entries for file are given.

File Organization

The main objective is to see how data is organized for storage and retrieval purposes. File system manager manages the storing and retrieving scheme. It has full authority of accessibility and taking decisions.

The main important function of file system manager is to organize or systemize all the data in order to enhance total system's performance and efficiency. The type of storing and retrieval scheme is decided by configuration of computer system. In normal operation, files can be systemized or organized by two ways:

(a) Sequential file organization

(b) Direct file organization

(a) Sequential File Organization

The name suggests, the basic operation of file organization. Sequential means one after another. The records of files are stored in physical order one after other as you created. The order is same for writing and reading i.e. you have no choice to change reading order of records. This scheme is suitable for electromagnetic storing devices like tapes and so on.

Advantages:

(1) The next record physically resides after the previous one. So separate pointer is not required to keep the track of records while reading them.

(2) Unidirectional access is provided

(3) This is device independent, so any of devices can be used to organize sequential files. Automatically, portability enhances.

Disadvantages:

If you want any record, then you must always start from first record. When desired record is found you can read record. You cannot directly access a particular record so it is time-consuming scheme. This leads to non-optimal utilization of device and low access efficiency.

(b) Direct file organization

In order to overcome the main disadvantage of large access time and inefficiency in sequential file organization the concept of direct file organization is developed. Direct file organization accesses the records through record's physical addresses on a direct access storage device.

Advantages:

(1) Any record in the file can be accessed randomly.
(2) Access time is minimized.

Disadvantages:

(1) As the physical address is to be generated, automatically processing time increases.
(2) The mechanical movement of I/O device is more.
(3) Possibility of poorer utilization.
(4) Can not be stored on sequential access devices like (magnetic tapes) i.e. it is not device independent.

Device Driver

Device drivers are software programs required for each device. Each device will require different driver for different device as per functionality. A device driver knows how the buffers, flags, register control and status bits should be used for a particular device. For simply reading a character from a device involves complex sequences of device specific operations.

Some device drivers are useful for data conversion. Devices deal with 8 bit or 16 bit data but computer requires 32-bit/ 64-bit data bus. Then device driver enables this conversion from one form to another.

Terminal I/O

Terminal hardware can be divided into two parts, the keyboard which is used for input medium and video screen, which is useful as an output medium. Combination of these two is termed as VDU.

Terminals are of two types,
1) Dumb terminal
2) Intelligent terminal.

1) Dumb terminal

It can have a microprocessor inside it and limited memory. It is responsible for basic input and output of characters. It is called dumb because it does no processing on the input characters.

2) Intelligent terminal

The terminal can do processing on the μp. So it requires more powerful hardware and software for it.

Process

Process is a program under execution, which competes for the CPU time and other resources. Generally, program exists on a disk or a paper. It is compiled but it does not compete for CPU time. But whenever a user wants to execute a program, it will be loaded in memory. At that time it becomes a process.

1.6 MULTIPROGRAMMING

In earlier days the computer systems were very costly and CPU's idle time was more. For increasing CPU utilization and for reducing idleness of CPU multiprogramming is used. It means several processes appear to run simultaneously.

Degree Of Multiprogramming

The no. of processes running simultaneously competing for CPU is called degree of multiprogramming.

Context Switching

In multiprogramming at a time multiple processes are running simultaneously. Suppose there are two processes, process 1 and process 2.

Then there would be time lost in turning attention from process 1 to process 2. That is called context switching.

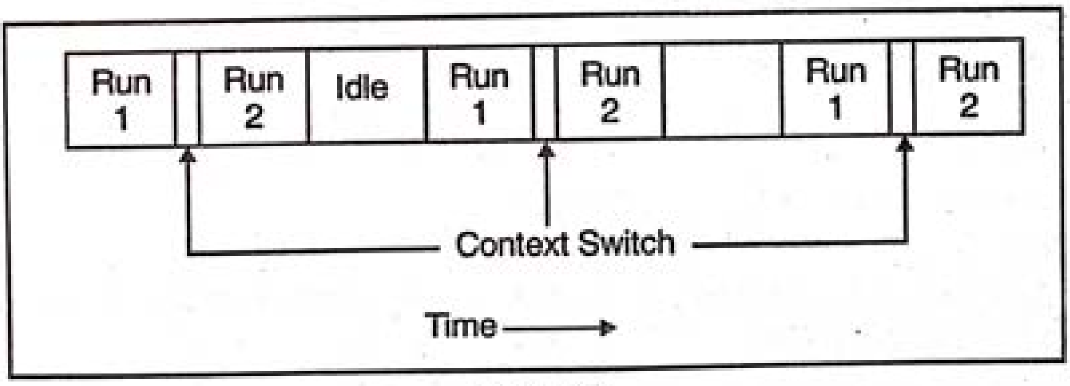


Fig. (1.7)

Context means storing the pointers of the memory contents of process and all CPU registers like PC, IR, Acc., SP and other general purpose registers. Each process has a register save area, which the operating system maintains, one for each process. In that area context of a process is saved.

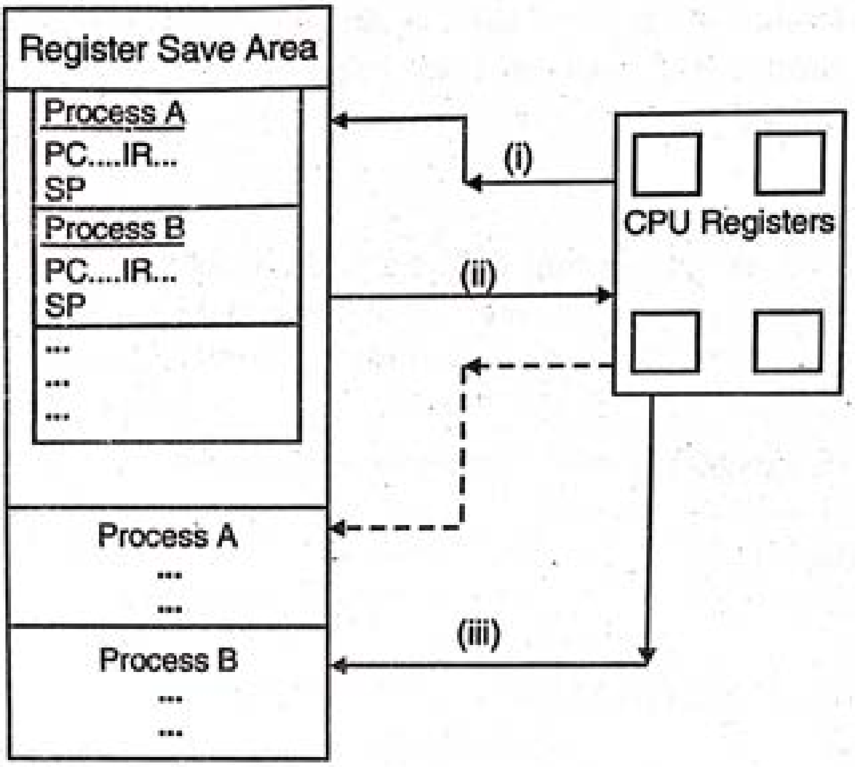


Fig.(1.8) Context switching (From Process A to B)

1) Refer fig. (1.8) Process A is running
2) Suppose A requires I/O operations then process B should start. For that context switching is required. At that time operating system stores the CPU registers for process A (i in fig) and restores the already saved contents of process B in CPU registers (ii in fig.).
3) When process A is scheduled again process B registers are stored and registers for process A are restored in CPU and it continues (iii in fig.) where it had left from.

Process States

Operating system defines three basic process states as given below.

1) **Running:**

A process, which is executing is termed as running process. In multiprocessing there will be many running processes and operating system will have to keep track of all of them.

2) **Ready:**

A process, which is not waiting for any external event such as an I/O operation, is said to be in a ready state.

3) Blocked:

Process, which is waiting for an external event such as an I/O operation, it is said to be in a blocked state.

The major difference between a blocked and a ready process is that a blocked process can't be directly scheduled even if CPU is free whereas ready process can be scheduled if CPU is free.

1.7 SCHEDULING

While scheduling various processes there are many objectives for operating system to choose from. Some of the objectives are listed below.

1) Fairness
2) Good throughput
3) Good CPU utilization
4) Low turn around time
5) Low waiting time
6) Good response time

(1) Fairness:

It refers to being fair to every user in terms of CPU time it gets.

(2) Throughput:

It refers to the total productive work done by the users put together.

(3) CPU Utilization:

It is the fraction of time the CPU is busy on an average with the user processes or the operating system. If the time slice is very small, then the context switches will be more frequent hence CPU will be busy executing O/S instructions rather than those of user processes. Therefore throughput will be low but CPU utilization will be very high.

(4) Turnaround Time:

It is the elapsed time between the time a program or job is submitted and the time when it is completed.

(5) Waiting Time:

It is time job spends waiting in a queue of the newly admitted processes in the O/S to allocate resources to it before commencing its execution.

1.8 PRIORITY

When no. of processes are competing for the same available resources like CPU and memory, then the concept of priority becomes useful. The priority can be external or internal.

External Priority

The user specifies it externally at the time of initiating the process. If user does not specify any external priority operating system assumes a certain priority called allocated a very short period of CPU time one by one. The short period of time during which a user gets the attention of CPU is known as a time slice.

The processing speed of system and use of multiprogramming in conjunction with timesharing allows the CPU to switch from one user station to another and do a part of each job in the allocated time slice until the job is completed.

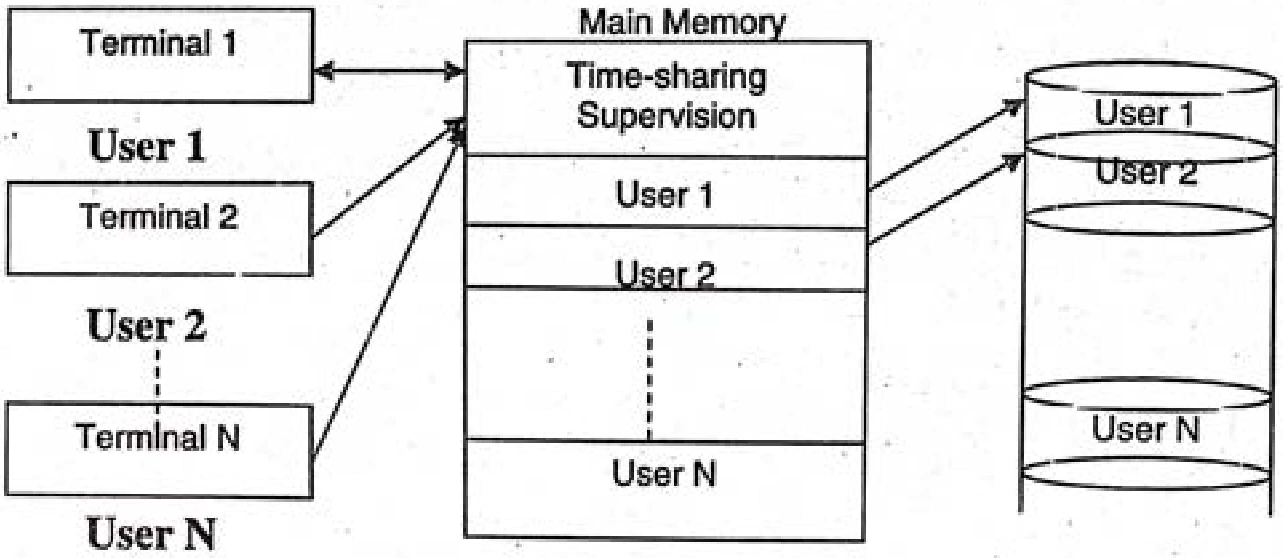


Fig. (1.9) Idea of Timesharing

Though it appears several users are using the computer system at the same time, only one program can be in control of CPU. So all the users will fall in one of the following three status groups,

1) Active: The user's program currently has control of the CPU.
2) Ready: The user's program is ready to continue but waiting for turn to get the attention of CPU.
3) Wait: The user waiting for some I/O operation.

Timesharing system reduces CPU idle time. It increases CPU utilization also.

1.12 MEMORY MANAGEMENT

Under this service generally memory allocation, deallocation is done by operating system. Again free memory is also checked.

For above things to be done variety of memory management systems are there as shown in figure.

- ***Contiguous, Real Memory Management System***
  - Single Contiguous, Real Memory Management System
  - Fixed Partitioned Memory Management System
  - Variable Partitioned Memory Management System
- ***Non - Contiguous, Real Memory Management System***
  - Paged Memory Management System
  - Segmented Memory Management System
  - Combined Memory Management System
- ***Non - Contiguous, Virtual Memory Management System***
  - Virtual Memory Management System

Fig. (1.10)

In contiguous scheme program is loaded in contiguous memory locations. But noncontiguous doesn't require it. The program should be divided into chunk. Chunks are of same size in paging and different size in segmentation. Again memory can be real memory or virtual memory. In real memory full process image is expected to be loaded in memory while in virtual memory only a part of process image is loaded in memory. We are focusing above all those terms in detail in this chapter.

Single Contiguous Memory Management

In this scheme of single contiguous memory management physical memory is divided into two contiguous areas. One of them is permanently allocated to the resident portion of the operating system (monitor) as shown in fig. (1.8). The operating system may be loaded at the lower addresses to top as shown in fig. or it can be loaded at the higher addresses. This choice is normally based on where vectored interrupt service routines are located as these addresses are determined at the time of hardware design in computers.

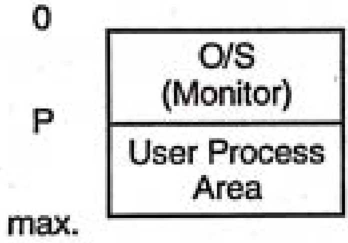


Fig. (1.11) Single contiguous memory management

At any time only one user process is in memory. Whenever this process is completed then the next process is brought into memory.

a) All ready processes are held on disk in an exe form. OS holds their PCBS in memory according to their priority.
b) Only one process runs in memory.
c) When that process is blocked, it is swapped out from memory to disk.
Next priority process swapped in the main memory from disk and it starts running.

This scheme has very fast access time but use is limited, as it is not having multi-user facility.

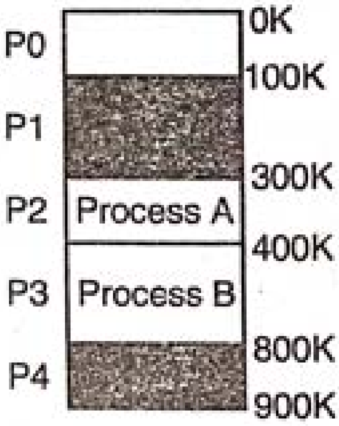
Partitions

Generally main memory is divided into various sections called partitions. In this scheme two types of partitions are available I) Fixed partitions II) Variable partitions.

Fixed Partitions

In this scheme partitions could be of different sizes. Once decided at the time of system generation they could not be changed.

System generation is the process of tailoring the operating system. It consists of number of routines for supporting a variety of hardware items and devices. According to devices used the device routines are selected at the time of system generation.



Partition ID	Partition		
	Starting Address	Size	Status
0	0	100	ALLC
1	100	200	FREE
2	300	100	ALLC
3	400	400	ALLC
4	800	100	FREE

Fig. (1.12) Fixed Partitions and its table

If you want to change partition, operations should be stopped and operating system has to be reloaded with different partition specification. Whenever partitions are declared operating system creates a partition description table (PDT) for future use as shown in fig. Initially all the entries are marked as free. When a process is loaded into the partition, that partition is marked as ALLOCATED refer fig.(1.12).

- Drawback of this scheme is fragmentation.

Internal Fragmentation

If a partition is of 100K and has to be allocated to a process of 80K; then 20K memory is wasted. This is called internal fragmentation.

External fragmentation

•If two free partitions of 30K and 50K are available and a process of 60K has to be accommodated. Then both the partitions can't be allocated in this process. So there is wastage of memory space. This is called external fragmentation.

Time complexity is very low because allocation/ de-allocation routines are simple as the partitions are fixed. Access time is not very high.

Variable Partitions

Due to some drawbacks of static partitions like fragmentation and restriction on no. of resident processes, (which affects degree of multiprogramming and so CPU utilization also.) variable partitions come into existence. In variable partitions no of partitions and their sizes are variable.

At any time any partition of memory can be either free or allocated to some process. But in this scheme starting address of any partition is varying.

Refer fig. (1.13) the scheme working is as follows;

1) Operating system is loaded in memory; all the rest of memory is free.
2) Program P1 is loaded in the memory and it starts executing.
3) Program P2 is loaded in the memory and it starts executing.
4) Program P3 is loaded in the memory and it starts executing.

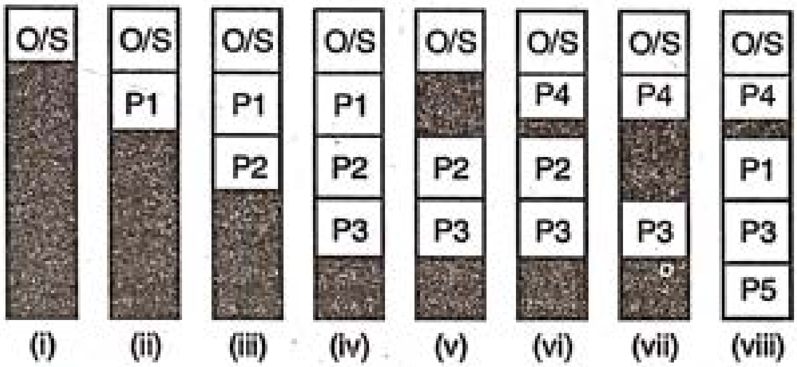


Fig (1.13): Memory allocation changes with 8 events

5) P1 is blocked. After a while P4 wants to occupy the memory. The existing free space is less than the size of P4. Assume that P4 is smaller than P1 but bigger than free area available at the bottom. So P1 is swapped out and two chunks of free space in memory.
6) P4 is loaded and it starts executing. P4 is loaded in the same space available as P4<P1
7) P2 terminates. Only P4 and P3 continue. The free area at the top and the one released by P2 can be joined.
8) P1 is swapped out. Also free space in middle is sufficient to hold P1. Another process P5 is loaded in the memory.

Thus it starts with two partitions and at stage 8 there are 6 partitions. These partitions are created by the operating systems at run time and they differ in sizes.

This scheme wastes less memory than the fixed partitions because there is no internal fragmentation.

Time complexity is certainly higher than fixed, due to various data structures and algorithms used in this method.

1.13 NON-CONTIGUOUS ALLOCATION

In various contiguous memory allocation schemes a problem of fragmentation arises, so non-contiguous allocation provides a better method to solve these problems.

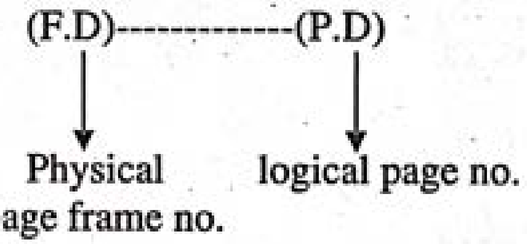
For that following various methods are used.

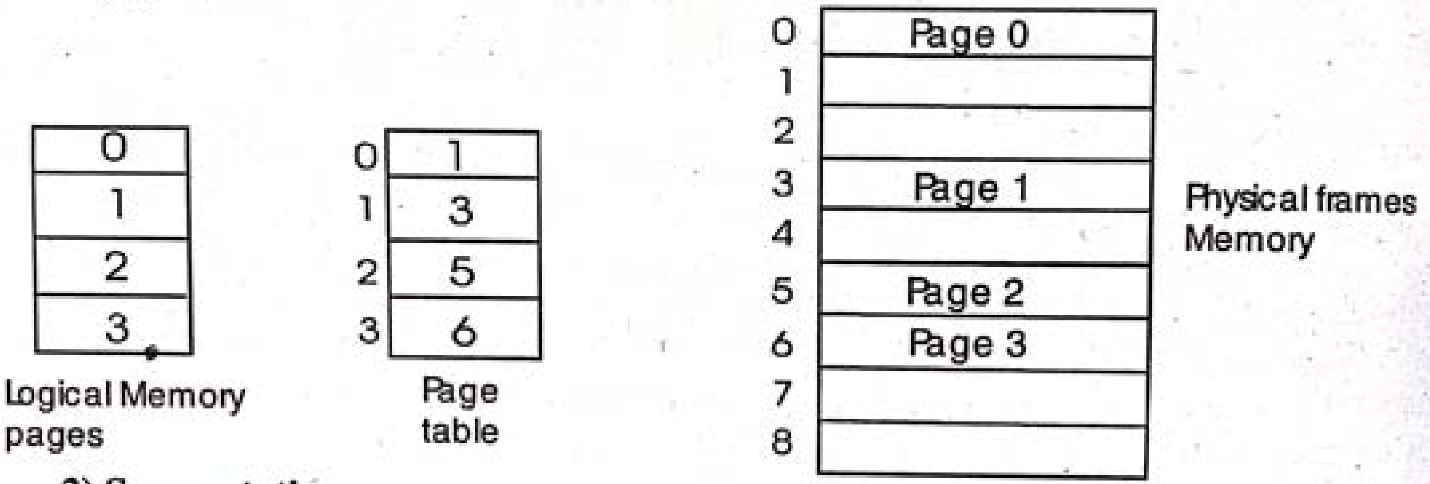
1) Paging

In paging the chunks of memory are of same size. The logical or virtual address space of program is divided into equal size pages and physical main memory is divided into equal sized page frames. The size of page is the same as that of a page frame, so that a page can exactly fit into a page frame. Therefore it can be assigned to any page frame, which is free.

The addresses which program refers are called virtual or logical addresses. In reality the program may be loaded at different memory locations are called physical addresses.

In this scheme generally virtual addresses consist of two parameters, a logical or virtual page no., and displacement D within the page. During execution address translation has to be done i.e. Virtual page no. should be converted into physical page no.

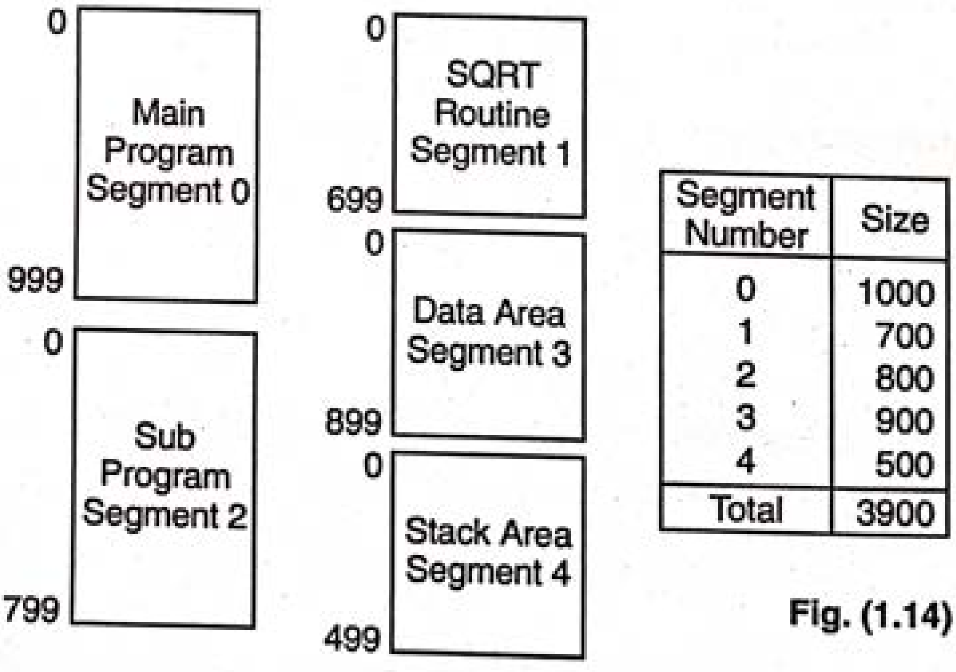

(F.D)--------------(P.D)
Physical page frame no.
logical page no.


0
1
2
3
Logical Memory pages
0 1
1 3
2 5
3 6
Page table
0 Page 0
1
2
3 Page 1
4
5 Page 2
6 Page 3
7
8
Physical frames Memory

2) Segmentation

Segmentation is similar to paging. Pages are physical in nature and hence are of fixed size but segments are logical division of a program so, of variable size.

Each program in its exe form can be considered to be consisting of major segments, code, data and stack. Each of these can be divided into further segments. Generally each program has a main function and few subprograms. A program can use various functions also. The fig.(1.14) shows various segments along with their sizes. Each segment is compiled with respect to 0; as the starting address for that segment.


0
Main Program Segment 0
999
0
Sub Program Segment 2
799
0
SQRT Routine Segment 1
699
0
Data Area Segment 3
899
0
Stack Area Segment 4
499

Segment Number	Size
0	1000
1	700
2	800
3	900
4	500
Total	3900

Fig. (1.14) Segmentation

1.14 VIRTUAL MEMORY

Virtual memory systems can be implemented using paging, segmentation or combined schemes. The Process can start executing with only part of the process image in the memory. In real memory systems entire process image has to be in memory. Actually this scheme is simple, as whole process has to be swapped in or out. But if the physical memory is limited, then the no. of processes it can hold at any time becomes limited. So virtual memory systems are implemented. The idea is when a page not currently in memory is referenced then only that page can be brought in to the memory.

If there is no page frame free in the physical memory for new page then O/S has to overwrite an existing page in physical memory. Which page should be overwritten is decided by page replacement policy. There are some terms related to virtual memory system are as follows

Locality of reference

This gives some basis to forecast whether a page is likely to be referenced in the near future based on its passed behavior and hence also whether you could throw that page out to make room for a new one. Clustering of page references in a certain time zone is called the principle of locality of reference.

Page Fault

When a process is executing with only a few pages in memory, and when an instruction is encountered which refers to any instruction or data in some other page, which is outside the memory, a page fault occurs.

Working Set

At any time a process has a no. of pages in the physical memory. The set of pages in the physical memory actively referred to at any moment is called working set.

Page Replacement Policy

As no. of processes and no. of pages in main memory for each process increase; at some point of time, all the page frames become occupied. At this time, if a new page is to be brought in, the O/S has to overwrite some existing page in memory. The page to be replaced is selected by page replacement policy. O/S designer chooses these type of policy.

Dirty Page

Before overwriting a page in the memory the operating system has to check if that page has been modified after it was loaded from the disk. If that page is modified, it becomes a dirty page. The operating system maintains one bit for each physical page frame for checking whether a page is dirty or not. This bit is called dirty bit.

Demand paging

In demand paging a page is brought in only when demanded.

1.15 BASICS OF GUI

Poor human computer interface has always been a problem area in using computers. In the past computers were not very powerful. Cryptic commands provided an interface to the computers operating system.

The users were expected to remember these commands and what they stood for. All the users would not be computer literates unless they knew the cryptic human – computer interface. The interface had to become more users friendly. Cryptic commands were converted into graphical representation. Such an interface is called graphical user interface or GUI.

Windows operating system is GUI based operating system. The screen can be split into different partitions. Each partition can be of different size. This partition is called window and hence called windowing technology.

Some of the characteristics of a window are its title, borders, work area and command area.

Components of a Window

Menu bars normally appear at the top of window under window title. When menu option is selected a pull down menu appears on the screen. A pull down menu will have an action on the left side and keyboard accelerates combination on the right.

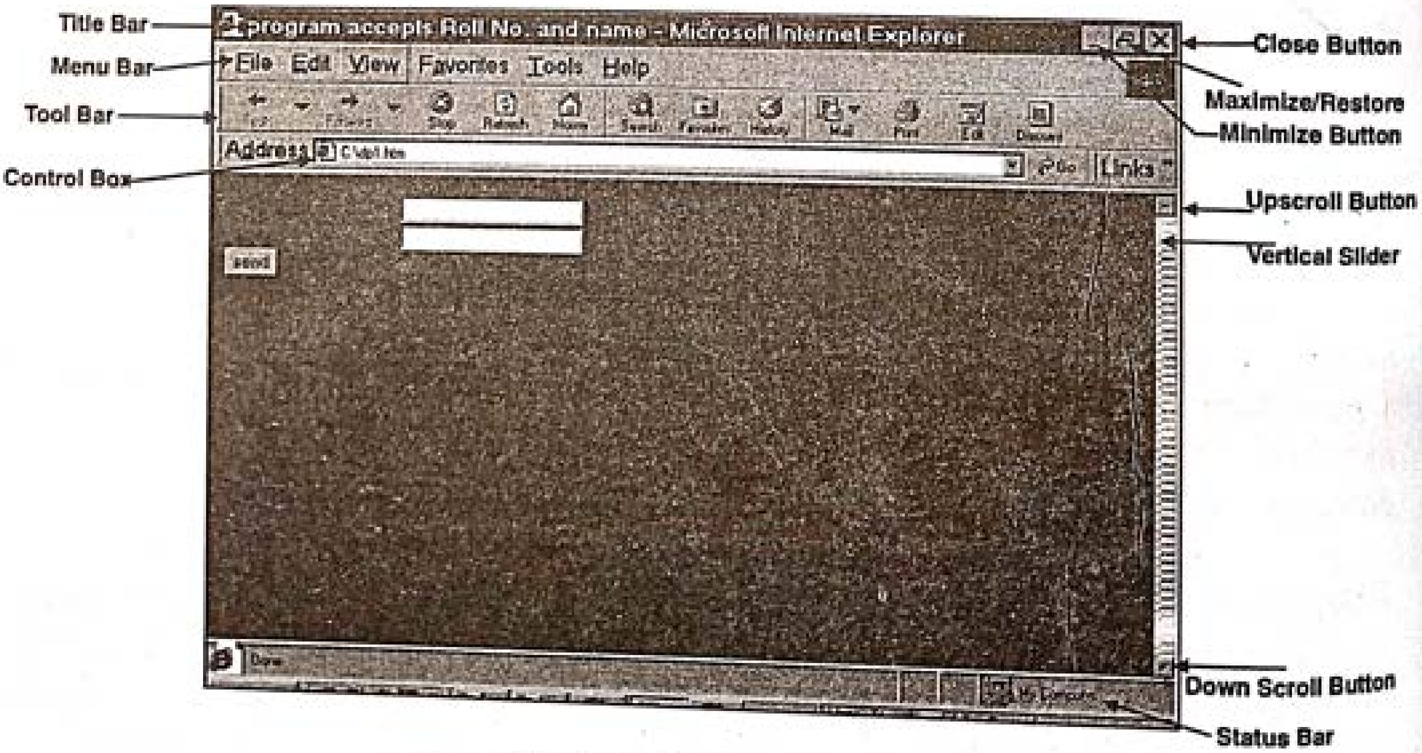

Title Bar
Menu Bar
Tool Bar
Control Box
program accepts Roll No. and name - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Address
Links
Close Button
Maximize/Restore
Minimize Button
Upscroll Button
Vertical Slider
Down Scroll Button
Status Bar

Fig. (1.15) Typical Window Structure

Scroll bars

A scroll bar consists of a horizontal or vertical scroll area with a slider box and an arrow in a box at each end. They are used to look at information, which is not currently visible in a window.

Title bar

It helps to identify each window separately. It displays generally programs name in title bar.

Every Window is having minimize, maximize, restore button.

Minimize

This is used to minimize program window; but program is not closed, it goes to bottom line.

Maximize/restore

It is used for making window full screen. Then this button becomes restore button. When you click this button, window comes to its previous size.

Close It is used for closing window.

Dragging Window

For changing the position of window on screen dragging is used. For that select a window by clicking it. Press left mouse button. Keeping in that way move mouse pointer to new position on screen. Then release mouse button, now window appears at new position.

Resizing Window

Changing the size of window is called resizing. For resizing move the pointer on to the border of window. Now mouse pointer shape changes to double arrow, press left mouse button and move the pointer to left or right (according to border whether left or right.) and release button. Now the window size is changed.

1.16 ACCESS AND SECURITY ASPECT OF O/S

In general security systems will control through use of specific security features, access to information that only properly authorized individuals or processes operating on their behalf will have access to read, write, create or delete.

OSI defines elements of security in the following terms:

1) Confidentiality: Information is not accessed in an unauthorized manner. (Read)
2) Integrity: Information is not deleted in unauthorized manner. (Write)
3) Availability: Information is available to authorized users at right time.

Security

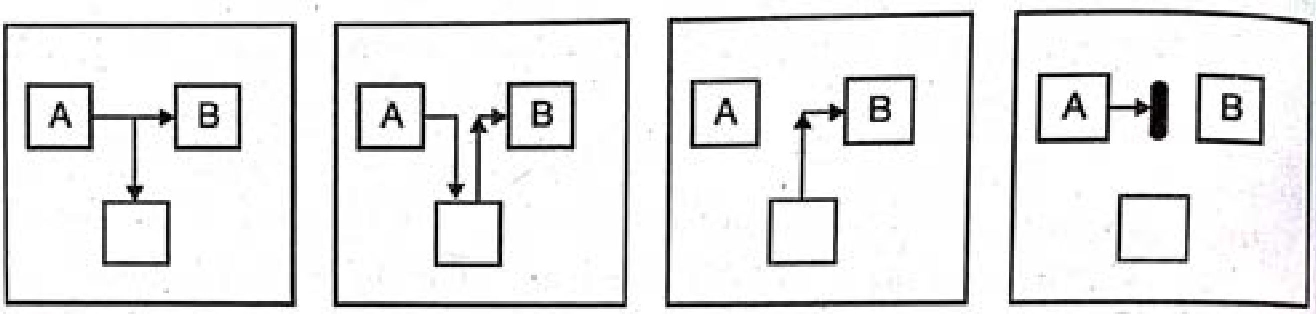
Ability of O/S to control storage and transportation of data in and between the objects O/S supports is security. In multi-user environment concepts of security are very important.

Security Threats

1) Due to networking the user has to access data and programs at different locations. This has increased the threat to the security of computing.
2) Sharing increases security threats.

Major Threats

1) Unauthorized use of service (tapping)
2) Unauthorized disclosure of information (disclosure)
3) Unauthorized alteration and deletion of information (amendment)
4) Unauthorized fabrication of information.
5) Denial of service to authorized users.



Tapping / Disclosure Amendment Fabrication Denial

Fig. (1.16)

Tapping and disclosure are passive threats and remaining are active threats.

Attacks on security

Security can be attacked in following ways.

1) **Authorization**

It means verification of access to the system resources.

1) Intruder may guess or steal password and use it.
2) Intruder may use vendor-supplied password, which is expected to use by system administrator.
3) Intruder may find password by trial and error method.
4) If user logs on and go for a break then intruder may use terminal.
5) An intruder can write a dummy login program to fool user and that program collects information for its use later on.

2) **Browsing**

Files are very permissive so one can easily browse system files. Due to that it may access database and confidential information can be read.

3) **Trap doors**

Sometimes software designers want to modify their programs after installation. For that there are some secret entry points which programmers keep and it does not require any permission. These are called trap doors. Intruders can use these trap doors.

4) **Invalid Parameters**

Due to invalid parameters some security violation can take place.

5) **Line Tapping**

Tappings in communication line can access/modify confidential data.

6) **Electronic Data Capture**

Using wire taps or mechanism to pick up screen radiation and recognize what is displayed on screen is termed as electronic data capture.

7) **Lost Line**

In networking, line may get lost. In such case some O/S log out and allow access only after correct identify of user. Some O/S cannot do this. So process will be floating and allow intruder to access data.

8) **Improper Access Controls**

Some administrator may not plan about all rights. So some users may have more access and some users have very less access.

9) **Waste Recovery**

If block is deleted its information will be as it is, until it is allocated to another file. Intruder may use some mechanism to scan these blocks.

10) **Rogue Software**

Programs are written to create mischief. Some of the programs under this are as follows:

a) Trojan Horse: These programs seem to be harmless but actually harmful.
b) Chameleon: It is similar to Trojan horse. It is a mimic logon program to collect all valid usernames and passwords on a system.
c) Ordinary Software Bomb: It explodes as soon as it is executed.
d) Timed S/W Bomb: It becomes active only at a specific time.
e) Logical S/W Bomb: It is activated only if logical condition is satisfied.
f) Worms: Programs attacking on node and spreading on other nodes.
g) Virus: It gets attacked to other programs to cause damage.
h) Rabbits: They are like worms. They replicate as soon as they execute.

1.17 COMPUTER WORM

A computer worm is a full program by itself. It is written in such a way that it spreads to other computers over a network and it also consumes all resources of network. It was invented by research scientists at XEROX PARC research center. Internet Worm was introduced on November 2, 1998 into Internet by Robert Morris.

Safeguards against worms

a) Prevent its creation: This is achieved by strong security and protection policies.
b) Prevent it's spreading: Before transferring any file on a network, user should be forced to sanction transfer.

1.18 COMPUTER VIRUS

It is written with a clear intention of infecting other programs. It is a part of a program, which normally piggybacks on a valid program. But it is not a complete program by itself.

Types of viruses

There are several types of computer virus as follows:

a) Boot sector virus
b) Memory resident virus
c) File virus
d) Command processor virus
e) General purpose virus

Generally classification is done based on what is affected or where the virus resides.

Infection methods

a) Append: In this method viral code appends itself to the unaffected program.
b) Replace: In this method viral code replaces original executable program completely or partially.
c) Insert: In this method the viral code is inserted in the body of an executable code to carry out some funny actions.
d) Delete: In this case viral code deletes some codes from executable program.
e) Redirect: In this case the normal control flow of a program is changed to execute some other code.

Modes of operation

Virus works in number of ways. Generally developer of virus produces any interesting or useful program such as a game or utility. But his program has viral code embedded in it. Generally this program is developed under DOS. Then it is distributed to people.

Virus detection

Normally virus detection program checks integrity of binary files. It maintains a checksum on each file. At regular frequency detection program calculates checksum and matches with original one. If there is mismatch then that program may be infected.

Virus removal

There are some viruses whose bit pattern in the code can be predicted. The virus removal program scans the disk for the patterns of known viruses and on detection it removes them.

Virus prevention

For prevention of virus always buy legal copies of software. Again take frequent backups of data and run monitor programs frequently to detect virus.

Difference between Computer Virus and Computer Worm

Computer Virus	Computer Worm
1) It is not a complete program by itself.	1) It is a complete program.
2) Viruses cause direct harm to the system by corrupting codes as well as data.	2) Worm generally consumes system resources.
3) It cannot operate independently	3) It operates independently.

QUESTIONS

1. Select the correct alternative

(i) _____ terminal does no processing on input characters.

(a) Intelligent (b) Dumb (c) RS232 (d) None of these

(ii) The time required for read write head to move to correct track is — **(Mar.09).**

(a) Latency time (b) Rotational delay (c) Seek time (d) None of these

(iii) 'Terminate a process' is the system call available in _____ management.

(a) process (b) memory (c) information (d) file

(iv) Linux is _______ type of software. **(March 2020)**

(a) Public (b) Free (c) Shareware (d) Licence

(v) Operating system is a _____. **(March 2003)**

a) Hardware, b) Software, c) Input device, d) Output device

(vi) _____ is not an operating system. **(March 2004)**

a) UNIX b) LINUX) MS-DOS d) C++

(vii) ____________ are the operating system programs. **(March 2005)**

a) Application programs b) Users programs

c) Process management programs d) Antivirus programs

(viii) _____ is an operating system. **(March 2007,2016)**

a) C^{++} b) C c) VB d) LINUX

(ix) Terminate a Process is the system call available in _______ **(March 2008)**

a) Process b) Memory c) Information d) File (x)Data is instantly updated in case of ―― operating system. **(Mar.2010)**

a) Batch Processing b)Multi Processing c) Multi user d) Real time

(xi) Context Switching is a term related to ―― management. **(March 2016)**

a) Process b) Memory c) Information d) device

(xii) the time lost in turning the attention of processor from one process to another is called as ―― **(March 2017)**

a) circuit switching b) bandwidth c)context switching d) bracket switching

2. What is Operating system? Explain major services of OS. **(March 2009,2020)**

3. What are the features of Windows NT operating system? **(March 2016,17)**

4. Explain in brief the following programs of MS-Window. **(March 2003,07,09)**

a) Program manager, b) File manager, c) Control panel

5. What are the components of LINUX operating system? Explain any three features of it. **(March2004,2019)**

6. With reference to process management explain the terms: **(March 2003)**

a) External priority, b) Purchase priority, c) Internal priority, d) Time slice

7. Explain the three main functions performed by a memory management module of an operating system. State any four-memory management systems.**(Mar.03,17)**

8. Which are three major areas in which the operating system divides its services. Give examples. **(March 2002,08)**

9. What is virtual memory? Explain following terms with respect to virtual memory.

a) Page fault b) Dirty bit

10. Explain context switching at process level in multiprogramming system with example. **(March 2002)**

11. What objectives are considered while scheduling processes? Explain.

12. State the various steps involved in allocation of a partition in case of fixed partition memory management. **(March 2002)**

13. What is GUI? Explain components of GUI. **(March 2004,05)**

14. Explain the following with respect to GUI: **(March 2007)**

i) Scroll bar ii) Title bar iii) Minimize

15. Explain in short the function of menu bar and scroll bar components of GUI.

(March 2002, 08)

16. What is GUI? Write any two features of GUI. (Mar.2005)

17.What is VDU? Explain the following terms.

a) Dumb terminal b) Intelligent terminal (March 2002)

18.What is virus? State the various types of virus and the basis in which they are classified. (March 2002)

19. What is Computer Virus? State any four methods by which a virus can infect other programs. (March2005,06,17,20)

20.What is computer worm? Explain its mode of operation. (March 2003)

21.. Differentiate between the following. (March2004)

a) Computer worm and computer virus (March2009)

b) Fixed partition and variable partition.

22. Explain the following terms a) Multiprocessing b) Degree of multiprogramming

23. Explain the difference between contiguous and non-contiguous allocation.

24.Explain the single contiguous and fixed partitioned memory management systems with a suitable memory mapping diagram. (March2005.10)

25.State the various steps involved in allocation of a partition in case of fixed partition memory management. (March2006)

26. Explain the concept of virtual Memory. Explain any three terms used in virtual memory. (March 2004)

27. Explain the file system related to Information Management with file operations only. (March 2007)

28. What are the different functions performed by Memory Management in operating system? Draw a memory map of a single user computer. Explain types of partitioning in brief. (March 2004)

29. What is meant by a System Call? How it is used? How does an Application program (AP) use these calls during execution? (March 2006)

30. Explain the following terms in case of magnetic disk: (March 2006,10)

i) Tracks and Sectors ii) Seek Time

iii) Transmission Time iv) Latency / Rotational Delay

31. Explain virus detection, removal and prevention (March 2016)

Answers Q. (1)

i) Dumb ii) Seek iii)Process

iv)Free v)software vi)C++ vii) Process management programs

viii) LINUX ix) Process x)Real time xi)process xii) context switching

❑❑❑

2
DATA STRUCTURES

INTRODUCTION

Everyone knows, what is program? One of the tools for writing the programs is High Level Language. Almost all high level languages provide the facility of built-in data structures. Often, we use these tools without concern about their actual implementation, such as a welder can use a welding machine without knowing much about electricity. However, built-in data structures provide limited applications. So in order to get number of interesting and useful ways of structuring data, programmer must build various data structures in his program. In this chapter, we will study some useful data structures.

In order to store and retrieve individual data items, the accessing function are defined. Due to these functions, the collected data is organized.

For implementation of any data structure we should know:

(a) How much memory space is required
(b) How to access data by coded functions

Data is simply a set of values. Generally collection of data is organized into hierarchy of fields, records, and files.

e.g. If you consider student's information as follows:

Name	Roll No.	Physics	Chemistry	Maths
Sachin	1	91	90	92

In above e.g. name, roll no., physics, chemistry and maths are fields. The collection of actual values of fields forms a record. The collection of such type of records forms a file.

Primary Key

A certain field may uniquely determine the record in a file. Such a field is called primary key. In the above example Roll No. is a primary key.

The above organization of data into fields, records, files may not be complex enough to maintain and efficiently process certain collections of data. For this reason data is organized into more complex type of structures.

2.1 DATA STRUCTURE

The logical or mathematical model of a particular organization of data is called data structure.

The choice of data model depends on two considerations,

1) It must be rich enough in structure to mirror the actual relationships of the data in real world.

2) Structure should be simple enough that can effectively process the data whenever necessary.

Few examples of data structures are array, list, binary tree etc.

Data Structure Operations

The data in data structures are processed by certain operations like:

1)**Traversing:** For processing certain item in record, each record is accessed exactly once. (Visiting a record)

2)**Searching:** Finding location of a record with given key value or finding the locations of all records, which satisfy one or more conditions.

3)**Inserting:** Adding a new record to the structure.

4)**Deleting:** Removing a record from a structure.

5)**Sorting:** Arranging the records in some order.

6)**Merging:** Combining the records in two different sorted files into a single sorted file.

2.2 ALGORITHMIC NOTATIONS

An algorithm is a finite step-by-step list of well-defined instructions for solving a particular problem. The form for formal representation of an algorithm consists of two parts,

1) It tells the purpose of algorithm, identification variables, which occur in algorithm and lists, input data.
2) It contains list of steps that is to be executed.

There are certain conventions, which has to be followed in algorithms.

i) Step, control, exit

The steps of algorithm are executed one after another beginning with step1.

Sometimes control may be transferred to step 'n' of algorithm by statement 'goto step n' or using control structures.

The algorithm is completed with the statement 'Exit'.

ii) Comments

It is given in brackets either at beginning or end of step. It indicates the main purpose of step.

iii) Variables

Variable names are generally given in capital letters. Also variables used as counters or subscripts are in capitals.

Assignment Statement

It uses: = notation

e.g. MAX: = 5

Where MAX is a variable

iv) Input / Output

For input read statement is used
Read: Variable name
For output write / print statement is used.
Write: Messages / Variable names

Procedure

It is used for independent algorithm modules. The term procedure will be used to describe a certain type of sub algorithm.

Example of algorithm

An algorithm to find largest element in array

A nonempty array DATA with numerical values is given. This algorithm finds the location LOC and the value MAX of the largest element of DATA. The variable K is used as a counter.

```
Step 1[Initialize]        Set K: = 1, LOC: = 1 and
                          MAX: = DATA [1]
Step 2 [Increment Counter]    Set K: = K+1
Step 3 [Test Counter] If K > N then:

        Write LOC, MAX and Exit
Step 4 [Compare and Update]
        If MAX <DATA [K], then:
        Set LOC: = K and MAX: = DATA [K]
Step 5 [Repeat loop]   Goto Step 2
```

Control Structures

Generally three types of logic are used in algorithms that are as follows,

1) Sequence Logic 2) Selection Logic 3) Iteration Logic

1) Sequence Logic

Generally modules are executed in some sequence. The sequence is represented by means of numbered steps. Generally flow pattern is as follows,

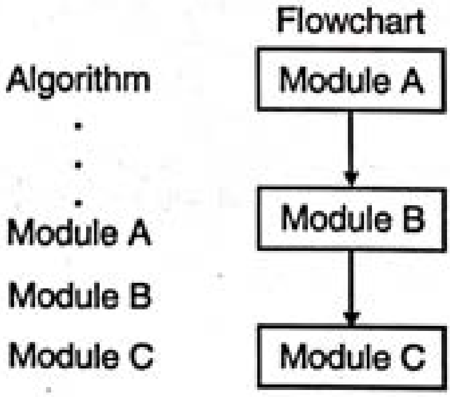


Fig. (2.1) Sequence Logic

2) Selection Logic

This logic employs a number of conditions. Accordingly which condition is satisfied that module is executed. Generally the following statement gives end of such a structure.

[End of IF Structure]

These structures are of three types

- **Single Alternative**

This structure has the form

```
IF condition, then:
        [Module A]
    [End of IF Structure]
```

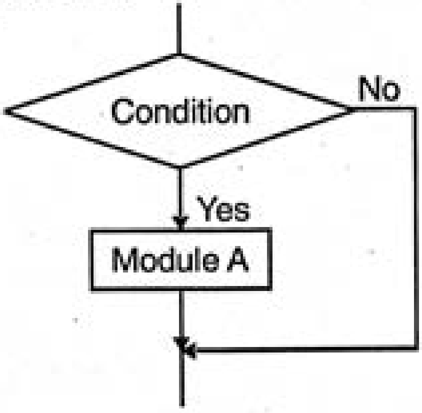


Fig. (2.2) Single Alternative

If condition holds true then Module A is executed, otherwise Module A is skipped and control transfers to the next step of algorithm.

- **Double Alternative**

This structure has following form

```
IF condition, then:
        [Module A]
Else:
        [Module B]
[End of IF Structure]
```

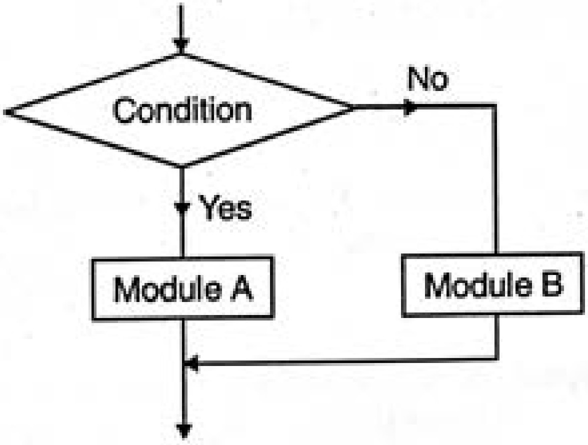


Fig. (2.3) Double Alternative

If condition is true Module A is executed and if condition is false Module B is executed.

Multiple Alternatives

```
If condition [1] then:
        [Module A1]
Else if condition [2] then:
        [Module A2]
        .
        .
        .

Else if condition [M] then:
        [Module AM]
Else:
        [Module B]
[End of if structure]
```

Example of Selection Logic;

Algorithm

This algorithm inputs the coefficients A, B, C of a quadratic equation and outputs real solutions if any,

```
Step 1 :    Read: A, B, C
Step 2 :    Set D: = $B^2 - 4AC$
Step 3 :    If D>0, then:
            a) Set X1: = $[-B + \sqrt{D}]/2A$
            and X2: = $[-B - \sqrt{D}]/2A$
            Else if D = 0, then:
            a)  Set X: = -B /2A
            b)  Write: 'UNIQUE SOLUTION'
            Else:
            Write 'NO REAL SOLUTIONS'
            [End of IF Structure]
Step 4:     Exit
```

3) Iteration Logic

Generally these types of structures are called loops. In that two types of loops are there, 1) Repeat for, 2) Repeat while

1) Repeat ... for

Form

```
Repeat for K = R to S by T
        [Module]
[End of loop]
```

Where K = Index variable
R = Initial value of K
S = Final value of K
T = Step (increment/decrement)

*The loop is executed for all values of K starting from R to S with step of T.

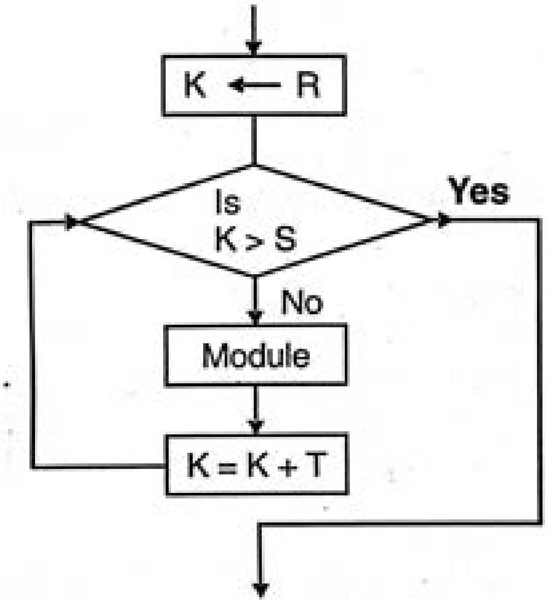


Fig.(2.4) Repeat ... For Logic

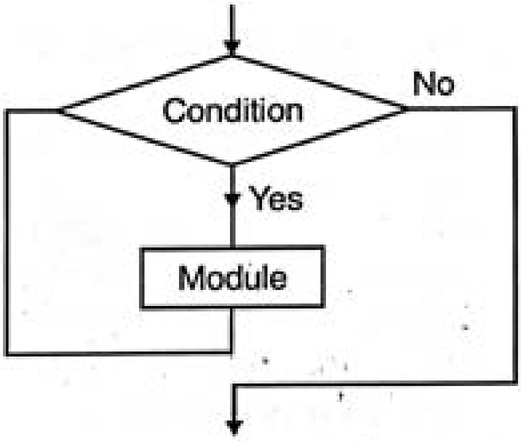


Fig.(2.5) Repeat.. While Logic

2) Repeat ... while
It uses a condition to control the loop.

Form

```
Repeat while condition:
        [Module]
[End of loop]
```

The loop continues till a condition is true. There must be initialization statement and there must be a statement that will change index variable in loop.

Algorithm using Repeat while loop (To find largest element in array.)

Given a nonempty array DATA with N numerical values. This algorithm finds the location LOC and the value MAX of the largest element of DATA.

```
1   [Initialize]   Set K: = 1, LOC: = 1
                   and MAX: = DATA [1]
2   Repeat steps 3 and 4 while K<= N
3   If      MAX<DATA [K] then:
            Set LOC: = K and MAX: = DATA [K]
            [End of If Structure]
4   Set K: = K+1
            [End of step 2 loop]
5   Write: LOC, MAX
6   Exit
```

Algorithm for inserting element into a linear array

```
INSERT [LA, N, K, ITEM]
LA = Linear array
N = Total no. of elements in the array
K = Any positive integer, K<=N
This algorithm inserts an element ITEM into the $K^{th}$ position in LA.
1.  [Initialize Counter] Set J: = N
2.  Repeat steps 3 and 4 while $J \geq K$
3.  [Move $J^{th}$ element downward]  Set LA [J+1]: = LA [J]
4.  [Decrease Counter]Set J: = J-1
        [End of step 2 loop]
5.  [Insert element]    Set LA [K]: = ITEM
6.  [Reset N]           Set N: = N+1
7.  Exit
```

Number	9 is inserted	4 is deleted
3	3	3
4	4	9
5	9	5
6	5	6
8	6	8
	8	
Original Array	Addition of element in middle array	Deletion of element from array

Fig. (2.6)

Deleting Element in array

It refers to the operation of removing one of the elements from A. Deleting at an end of array is simple but deleting an element somewhere in the middle of the array would require that each subsequent element be moved one location upward in order to fill up the array.

Algorithm for deleting an element

This algorithm deletes the K^{th} element from a linear array LA and assigns it to a variable ITEM.

```
DELETE [LA, N, K, ITEM]
LA = Linear array      N = Total no. of elements in array
K = Any positive integer such that K<= N

1. Set ITEM: = LA [K]
2. Repeat for J: = K to N-1
```

```
3. [Move J+1st element upward.]  Set LA [J]: = LA [J+1]
   [End of loop]
4. [Reset the number N of elements in LA.]
                  Set N: = N-1
5. Exit
```

2.4 BUBBLE SORTING

Sorting means, rearranging the elements in increasing or decreasing order. Suppose A[1], A[2],..... A[n] are in memory. Then bubble sort algorithm works as follows:

Compare A [1] and A [2] and arrange them in the desired order, so that A [1], A[2]. Then compare A [2] and A [3] and arrange them so that A [2], A [3]. Continue this process until we compare A [N-1] with A [N] so that A [N-1], A [N].

Step 1: It involves n-1 comparisons. During this step, the largest element is coming like a bubble to the n^{th} position. When step 1 is completed A[N] will contain the largest element.

Step 2: Repeat step 1 with one less comparison. Step 2 involves n-2 comparisons, when step 2 is completed we get second largest element A[N-2].

Step 3: Repeat step 1 with two less comparisons. It involves N-3 comparison.

Step N-1: Compare A[1] with A[2] and arrange them so that A[1] ,[A[2].

After n-1 steps the list will be sorted in increasing order.

Pass

The process of sequentially traversing through all or part of a list is called a pass. Each of above step is actually a pass. The bubble sort algorithm requires N-1 passes where n is number of input items.

The time required for sorting algorithm is measured in terms of comparisons. Specifically there are n-1 comparisons during first pass n-2 comparisons in second pass and so on.

Complexity of algorithm

$$\begin{aligned} f(n) &= (n-1) + (n-2) + \ldots.. + 2 + 1 \\ &= \frac{n(n-1)}{2} \\ &= \frac{n^2}{2} + 0(n) \\ &= o(n^2) \end{aligned}$$

The time required to execute bubble sort algorithm is proportional to n^2 where n is number of input items.

```
3. [Move J+1st element upward.] Set LA [J]: = LA [J+1]
   [End of loop]
4. [Reset the number N of elements in LA.]
                  Set N: = N-1
5. Exit
```

2.4 BUBBLE SORTING

Sorting means, rearranging the elements in increasing or decreasing order. Suppose A[1], A[2],..... A[n] are in memory. Then bubble sort algorithm works as follows:

Compare A [1] and A [2] and arrange them in the desired order, so that A [1], A[2]. Then compare A [2] and A [3] and arrange them so that A [2], A [3]. Continue this process until we compare A [N-1] with A [N] so that A [N-1], A [N].

Step 1: It involves n-1 comparisons. During this step, the largest element is coming like a bubble to the n^{th} position. When step 1 is completed A[N] will contain the largest element.

Step 2: Repeat step 1 with one less comparison. Step 2 involves n-2 comparisons, when step 2 is completed we get second largest element A[N-2].

Step 3: Repeat step 1 with two less comparisons. It involves N-3 comparison.

Step N-1: Compare A[1] with A[2] and arrange them so that A[1] ,[A[2].
After n-1 steps the list will be sorted in increasing order.

Pass

The process of sequentially traversing through all or part of a list is called a pass. Each of above step is actually a pass. The bubble sort algorithm requires N-1 passes where n is number of input items.

The time required for sorting algorithm is measured in terms of comparisons. Specifically there are n-1 comparisons during first pass n-2 comparisons in second pass and so on.

Complexity of algorithm

$$\begin{aligned} f(n) &= (n-1) + (n-2) + \ldots.. + 2 + 1 \\ &= \frac{n(n-1)}{2} \\ &= \frac{n^2}{2} + 0(n) \\ &= o(n^2) \end{aligned}$$

The time required to execute bubble sort algorithm is proportional to n^2 where n is number of input items.

Algorithm

[Bubble Sort] BUBBLE [DATA, N]

Here DATA is an array with N elements. This algorithm sorts the elements in DATA.

```
1. Repeat steps 2 and 3 for K = 1 to N-1
2. Set PTR: = 1 [? pass pointer PTR.]
3. Repeat while PTR<= N–K. [Executes pass.]
   a) IF DATA [PTR] > DATA [PTR + 1], then:
      Interchange DATA [PTR] and DATA [PTR + 1]
      [End of IF structure]
   b) Set PTR: = PTR + 1
      [End of inner loop.]
   [End of step 1 outer loop.]
4. Exit
```

Searching

Searching refers to the operation of finding the location of given element in the list. The most commonly used searching algorithms are linear search and binary search.

In linear search the given element is compared with each element of list one by one. This method is also called sequential search.

Algorithm Linear Search

LINEAR [DATA, N, ITEM, LOC]

Here DATA is a linear array with N elements and ITEM is a given item of information. This algorithm finds the location LOC of ITEM in DATA or sets LOC: = 0, if search is unsuccessful.

```
1. [Insert ITEM at the end of DATA] set DATA [N+1]: = ITEM
2. [Initialize counter] set LOC: = 1
3. [Search for ITEM]
   Repeat while DATA [LOC] ≠ ITEM
       Set LOC: = LOC + 1
   [End of loop]
4. [Successful?] IF LOC = N+1 then set LOC: = 0
5. Exit
```

2.5 BINARY SEARCH

While using this searching method array should be sorted in increasing numerical order.

Suppose DATA is an array, which is sorted in increasing numerical order, and we want to find out the location LOC of a given ITEM of information in DATA.

Then binary search algorithm applied works as follows. During each search for ITEM is reduced to a segment of elements of data:

DATA [BEG], DATA [BEG +1], DATA [BEG+2],..... DATA [END]

Here BEG and END variables denoted beginning and end locations of the segment under consideration. This algorithm compares ITEM with middle element DATA [MID] of the segment where MID is obtained by

MID = INT [(BEG + END)/2]

If DATA [MID] = ITEM then the search is successful and we set LOC: = MID otherwise a new segment of DATA is obtained as follows,

a) If ITEM<DATA [MID] then item can appear in left half of the segment.
   DATA [BEG], DATA [BEG +1],..... DATA [MID-1]
   So we reset END: =MID-1 and begin searching again
b) If ITEM > DATA [MID] then ITEM can appear only in the right half of the segment.
   DATA [MID+1], DATA [MID+2],..... DATA [END]
   So we reset BEG: = MID+1 and begin searching again.
c) If ITEM is not in DATA then eventually we obtain END < BEG

It means that search is unsuccessful. In this case we assign LOC: = NULL (Generally we begin with BEG=1 and END=N or more generally we can say BIG=LB and END=UB)

Algorithm for Binary Search

(Binary Search) BINARY [DATA, LB, UB, ITEM, LOC]

Here DATA is sorted array with lower bound LB and upper bound UB and ITEM is given item of information. The variables BEG, END and MID denote respectively the beginning, end and middle location of a segment of elements of DATA. This algorithm finds the location LOC of ITEM in DATA or set LOC=NULL.

```
1. [Initialize segment variables.]
       Set BEG: = LB, END: = UB and MID=INT [(BEG+END)/2]
2. Repeat steps 3 and 4 while BEG <= END and DATA [MID] ≠ ITEM
3. IF ITEM < DATA [MID], then:
       Set END: = MID-1
   Else:
       Set BEG: = MID+1
   [End of IF structure]
   Set MID: =INT [(BEG+END)/2]
   [End of step 2 loop]
   IF data [MID] = ITEM, then:
       Set LOC: =MID
   Else:
       Set LOC: =NULL
   [End of IF structure]
4. Exit
```

Here BEG and END variables denoted beginning and end locations of the segment under consideration. This algorithm compares ITEM with middle element DATA [MID] of the segment where MID is obtained by

MID = INT [(BEG + END)/2]

If DATA [MID] = ITEM then the search is successful and we set LOC: = MID otherwise a new segment of DATA is obtained as follows,

a) If ITEM<DATA [MID] then item can appear in left half of the segment.
   DATA [BEG], DATA [BEG +1],..... DATA [MID-1]
   So we reset END: =MID-1 and begin searching again
b) If ITEM > DATA [MID] then ITEM can appear only in the right half of the segment.
   DATA [MID+1], DATA [MID+2],..... DATA [END]
   So we reset BEG: = MID+1 and begin searching again.
c) If ITEM is not in DATA then eventually we obtain END < BEG

It means that search is unsuccessful. In this case we assign LOC: = NULL (Generally we begin with BEG=1 and END=N or more generally we can say BIG=LB and END=UB)

Algorithm for Binary Search

(Binary Search) BINARY [DATA, LB, UB, ITEM, LOC]

Here DATA is sorted array with lower bound LB and upper bound UB and ITEM is given item of information. The variables BEG, END and MID denote respectively the beginning, end and middle location of a segment of elements of DATA. This algorithm finds the location LOC of ITEM in DATA or set LOC=NULL.

```
1. [Initialize segment variables.]
       Set BEG: = LB, END: = UB and MID=INT [(BEG+END)/2]
2. Repeat steps 3 and 4 while BEG <= END and DATA [MID] ≠ ITEM
3. IF ITEM < DATA [MID], then:
       Set END: = MID-1
   Else:
       Set BEG: = MID+1
   [End of IF structure]
   Set MID: =INT [(BEG+END)/2]
   [End of step 2 loop]
   IF data [MID] = ITEM, then:
       Set LOC: =MID
   Else:
       Set LOC: =NULL
   [End of IF structure]
4. Exit
```

Complexity of Binary Search Algorithm

The complexity is measured by the number of f (n) of comparisons to locate ITEM in DATA where DATA contains n elements. Each comparison reduces sample size in half. So we require at most f (n) comparisons to locate ITEM where

$f(n) = [\log_2 n]+1$

Applications:

(1) This algorithm is used to search ordered array.

(2) To find out the target element whether it is present or not. If present then correspondingly it gives position in the array.

Limitations:

(a) The given list must be sorted.

(b) The access of list must be random means; the middle element can be accessed.

2.6 POINTER ARRAYS

An array whose each element is a pointer is called pointer array.

e.g. int * p[4] – pointer array
int a [4] – array of integers
p[0]= &a[0] p[1] = &a[1]
p[2]= &a[2] p[3]= &a[3]

This pointer array p holds the address of each element in array a so it is called pointer array.

Detailed Example of Pointer Array

Suppose TEAM is an array, which contains the locations of different teams or more specifically locations of first elements in different teams.

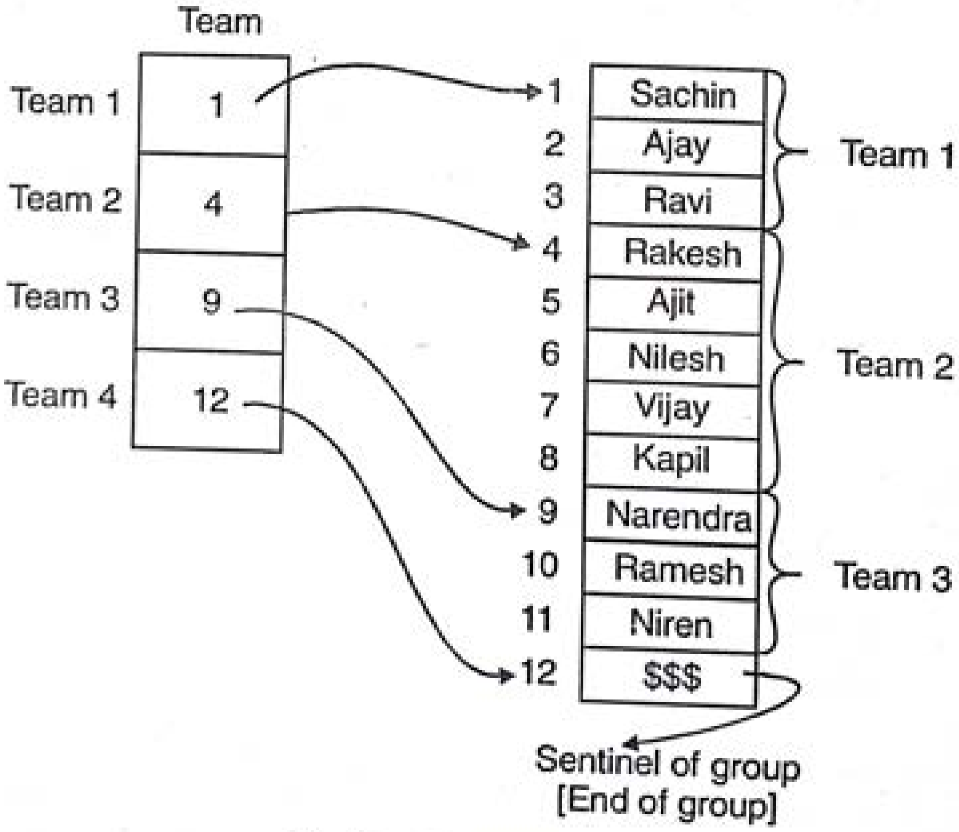

Team
Team 1
1
Team 2
4
Team 3
9
Team 4
12
1 Sachin
2 Ajay
3 Ravi
4 Rakesh
5 Ajit
6 Nilesh
7 Vijay
8 Kapil
9 Narendra
10 Ramesh
11 Niren
12 $$$
Team 1
Team 2
Team 3
Sentinel of group
[End of group]

Fig.(2.7) Example of an Array

Team [L] and Team [L+1]-1 contains respectively the first and last elements in Team L.

e.g. Team 1 → 1 → first element of Team 1
Team [1+1]-1 → 4 → last element of Team 1
Team 4 points to the sentinel of lists.

2.7 RECORD

A record is collection of related data items. Each data item is termed as field. File is collection of similar records. Each data item may be a group item composed of sub items.

Comparison

	Record	Linear Array
(1)	A record may be a collection of non homogeneous data; i.e. the data items in a record may have different data types.	The data items in an array may have same data types.
(2)	The data items in a record are indexed by attributes . So there may not be a natural ordering if its elements	There may be natural ordering of its elements.

Example of Record

Suppose a college keeps a record of each student, which contains the following data items.

ITEM	SUBITEM
Name	Last, First, MI (Middle Initials)
Address	
Stream	division
Phone	

The structure of above record is described as follows;

1. Student
   2. Name
      3. Last
      3. First
      3. MI (Middle Initial)
   2. Address
   2. Phone
   2. Stream
      3. Division

The number to the left of item name (i.e. identifying) represents a level number. Sub items follow each group item and the level of sub items is 1 more than the level of group item.

How to access item in record

Suppose we want to know the last name of student then it is accessed using dot operator as follows,

Student.name.last

Generally for separating sub items from group items dot is used.

Representation of records in memory

Since records contain nonhomogeneous data it can't be stored in array. Some higher-level languages are having built in record structure (for e.g. PL/1, PASCAL, COBOL). Suppose these types of structures are not available in language then record has to be stored in individual variables. But if we want to keep entire file of records, then all data elements belonging to the same identifier will be of same type. So a file may be stored in memory as collection of arrays.

e.g. Student file

Name	Address	Phone	Stream
Kulkarni Smita L.	19, Shaniwar Peth	4456184	Arts
Patil Ajit D.	24, M.G. Road	6361621	Commerce
.	.	.	.
.	.	.	.
.	.	.	.
	.		

Here elements in arrays name, address, phone, and stream, with same subscript belong to the same record.

2.8 LINKED LIST

Linked list or one-way list is a linear collection of data elements called nodes where the linear order is given by means of pointers. Each node is divided into two parts:

1. First part contains the information of the element.
2. Second part contains the address of next node in list (it is called link field).

START is a pointer variable, which contains the address of first node.

Left part represents the information part of the node while right part represents the next pointer field of node. The pointer of last node contains a special value called NULL, which is an invalid address.

The schematic diagram of a linked list with 4 nodes is shown below,

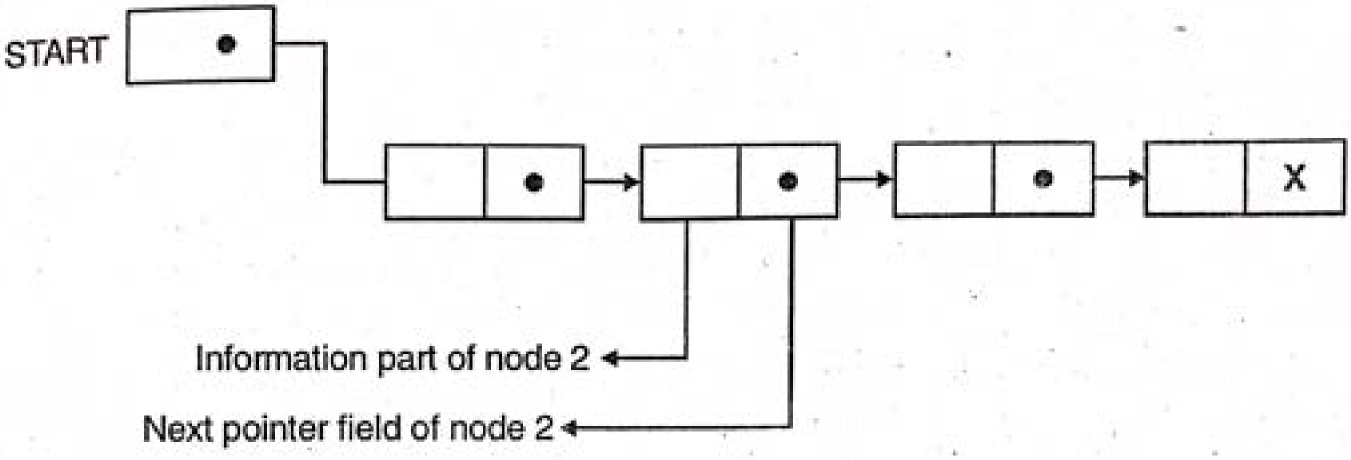
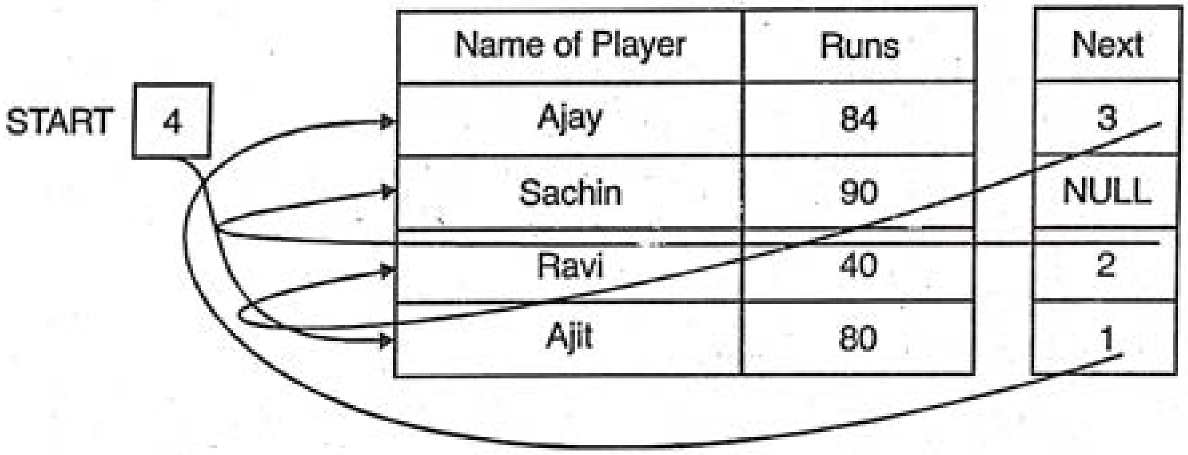


Fig. (2.8) A Linked List

The following diagram shows a linked list

1) Name of player and no. of runs from an information part while next pointer field gives the next node's address in linked list.

1) Start ⟶ 4
2) 4 ⟶ 1
3) 1 ⟶ 3
4) 3 ⟶ 2
5) 2 ⟶ last node



Name of Player	Runs	Next
Ajay	84	3
Sachin	90	NULL
Ravi	40	2
Ajit	80	1

Fig. (2.9)

Representation of Linked List in Memory

Let List be a linked list. Then LIST will be maintained in memory as follows,

1) LIST requires 2 arrays; we will call them here INFO and LINK such that INFO [K] and LINK [K] contain respectively the information part and next pointer field of a node K of list.
2) LIST also requires a variable name such as START, which contains the location of beginning of list and next pointer sentinel denoted by NULL, which indicates end of list.

The following e.g. shows how linked list is stored in memory. The nodes of the list need not occupy adjacent elements in the arrays INFO and LINK and more than one list may be maintained in the same linear arrays INFO and LINK. But each list must have its pointer variable giving the location of its first node.

START 9

	INFO	LINK
1		
2		
3	I	4
4	N	8
5		
6	I	7
7	S	12
8	K	11
9	L	3
10		
11	L	6
12	T	0

Linked List can be picturized in following manner:

1) START = 9, INFO [9] = L, LINK [9] = 3
2) INFO [3] = 1, LINK [3] = 4
3) INFO [4] = N, LINK [4] = 8
4) INFO [8] = K, LINK [8] = 11
5) INFO [11] = L, LINK [11] = 6
6) INFO [6] = I, LINK [6] = 7
7) INFO [7] = S, LINK [7] = 12
8) INFO [12] = T, LINK [12] = 0..... last node

i.e. "LINK LIST" string is stored in form of Linked List.

Advantages of Linked List

Generally lists are stored in form of arrays. But in arrays insertion and deletion is not easy. Also array size can't be easily increased. Using linked list this problem can be solved.

2.9 TREE DATA STRUCTURE

In case of Link data structure, each node has link information of next node. Thus sequentially, we can access all the nodes. It will be time consuming for searching long list;because each node has information about next node only.

In order to reduce time consumption, binary tree concept was developed. Before going to binary search tree, let us learn basics of tree structure.

Tree is an hierarchical structure of collected data items.It is nonlinear structure.

Node : It is as element of list. It may be a character, string or number.

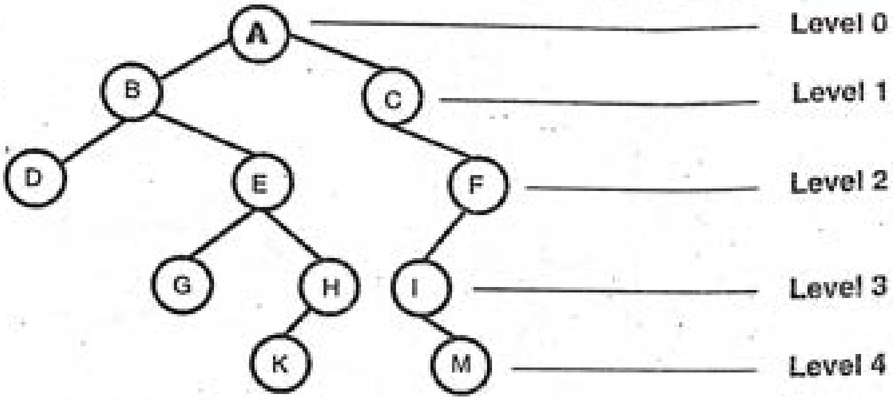


Fig. (2.10) A Tree data structure

Node		**Node's description**
ROOT	:	A node which has no parent. The node containing 'A' is the root of the tree.
CHILD	:	Nodes of root.
(a) LEFT CHILD	:	The left node of root.
(b) RIGHT CHILD	:	The right node of root.
		The node containing A has left child containing node B and right (child containing node C.)
PARENT		:Node which has child (or children). A is parent of B and C
LEAF		:The node which has no child (or children) D, G, K, M are leaves.
SIBLINGS	:	Two nodes have the same parent. The nodes containing D and E are both children of the node containing B. These nodes are siblings.
ANCESTOR	:	The node is an ancestor of another node, if it is a parent of that node. The ancestors of the node containing G are the nodes containing E, B, A. ROOT is always an ancestor of every node.
DESCENDANT	:	The node is a descendant of another node, if it is a child of that node. The descendant of the node containing E are the nodes containing G, H, K. All nodes in the tree are descendants of the ROOT.
LEFT SUBTREE	:	The descendants of root's left child which acts a root of subtree.
RIGHT SUBTREE	:	The descendants of root's right child which acts a root of subtree.
LEVEL	:	The distance from the ROOT. The ROOT is always at zero (0) level.

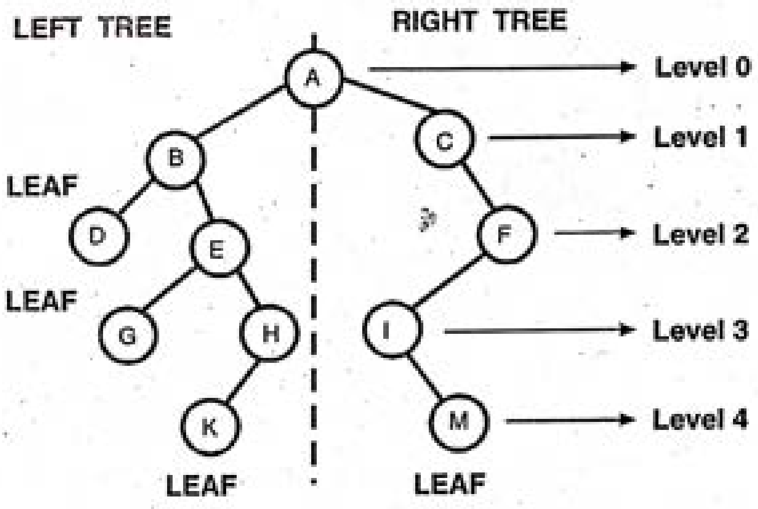


Fig. (2.11)

Types of Tree

Trees can be classified according to node structure such as

1) **Null tree** : a tree without node.
2) **Binary tree** : A tree which has left and right child.
3) **Ordered tree** : Children of each node are ordered from Left to Right.
4) **Nonordered tree** : Children of each node are not ordered from Left to Right.

Binary Tree

A binary tree T is defined as a finite set of elements called nodes, such that

1) T is empty (called NULL tree or empty tree) or 2) T contains a distinguished node R called the root of T and remaining nodes of T form an ordered pair of disjoint binary trees T_1 and T_2.

e.g.

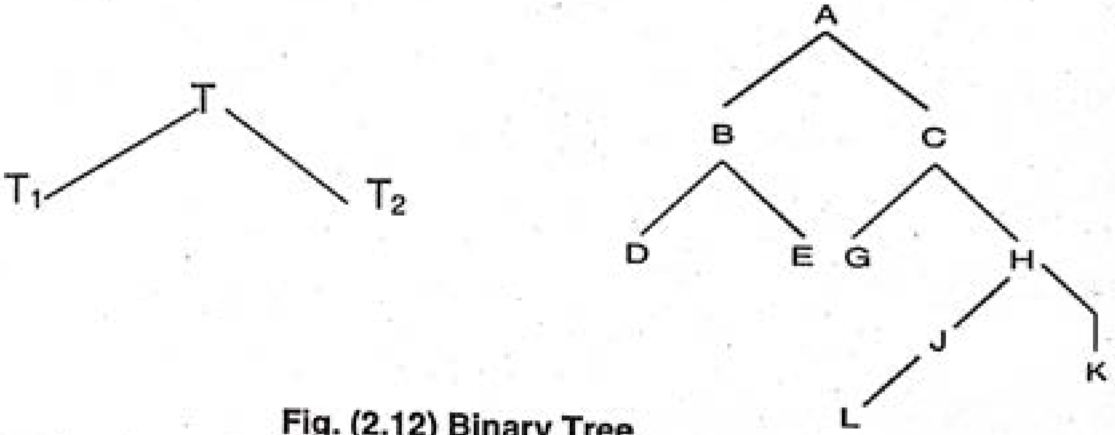


Fig. (2.12) Binary Tree

*If T contains a root R then the two trees T_1 and T_2 are called respectively left and right sub trees of R. If T_1 is nonempty, then its root is called left successor of R. Similarly if T_2 is nonempty then its root is called right successor of R.

1) B is left successor of A and C is right successor of node A refer fig. (2.12).
2) The left sub tree of root A consists of node B, D, and E and the right subtree of A consists of nodes C, G, H, J, K and L.
3) Any node N in a binary tree T has either 0, 1,2 successors. The nodes A, B, C, H have two successors. The nodes D, E, G, L and K have no successors that nodes are called terminal nodes.

Binary tree T and T_1 are said to be similar if they have same structure. The trees are said to be copies if they are similar and if they have same contents at corresponding nodes.

Algebraic Expressions

These can be represented as tree. Each variable or constant in E appears as internal node of T whose left and right subtrees corresponding to the operands of operation.

$$E = (a-b)^2/((c*d)+e)$$

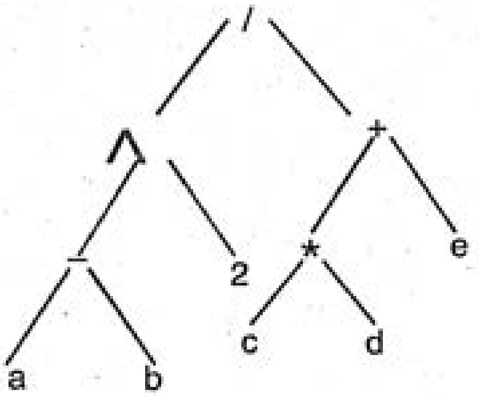


Fig. (2.13)

Complete Binary Tree

Consider any binary tree T. Each node at T can have at most two children. The level r of T can have at most 2^r nodes. The tree T is said to be complete if all its levels except possibly the last have maximum no. of possible nodes and if all the nodes at the last level appear as far as left as possible.

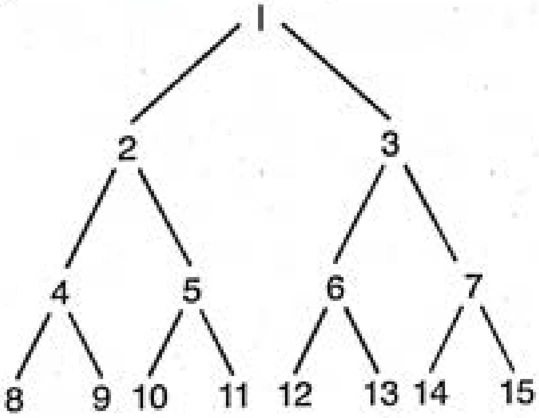


Fig. (2.14)

The left and right children of node K are 2 * K and 2 * K + 1 and the parent of K is the node K/2.

e.g. 4^{th} node

Parent node ⟶ 4/2 = 2

Children ⟶ 1) 4*2 = 8   2) (4 * 2) + 1 = 9

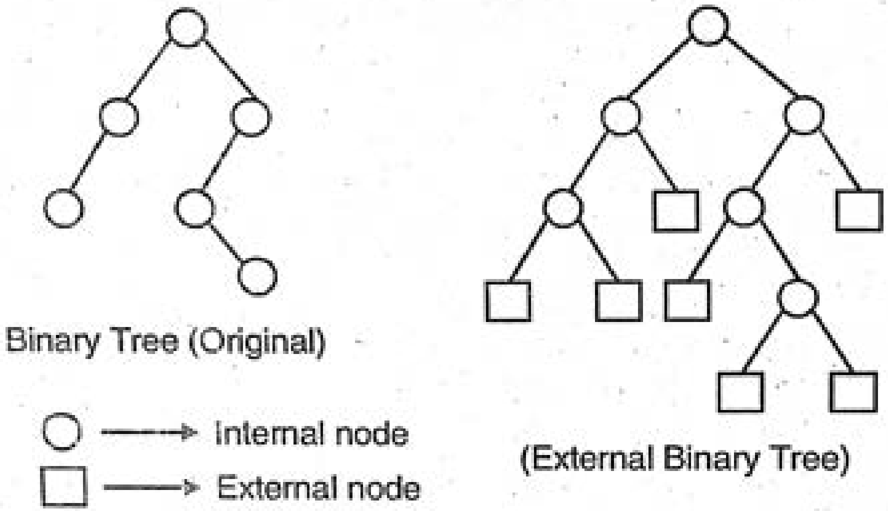
The depth of complete binary tree is given by,

$D_n = [\log_2 n + 1]$

Extended Binary Tree

A binary tree T is said to be 2 tree or extended binary tree, if each node N has either n or 2 children. In this case nodes with 2 children are called internal nodes while nodes with 0 children are called external nodes.

Any binary tree may be converted into a 2 tree by replacing each empty subtree by a new node. The nodes in original tree are internal nodes in the extended tree and the new nodes are external nodes in the extended tree.



Binary Tree (Original)

(External Binary Tree)

Fig. (2.15)

Representing binary trees in memory

Let T be a binary tree. There are two way of representing binary trees in memory. 1) Linked representation 2) Sequential representation

1) Linked representation

In this method 3 parallel arrays are used INFO, LEFT and RIGHT.

1) INFO [K] contains the data at node N.
2) LEFT [K] contains the location of left child of node N.
3) RIGHT [K] contains the location of right child of node N.
4) ROOT will contain the location of root R of T. It is a pointer variable.
5) If any subtree is empty then corresponding pointer will contain a null value.

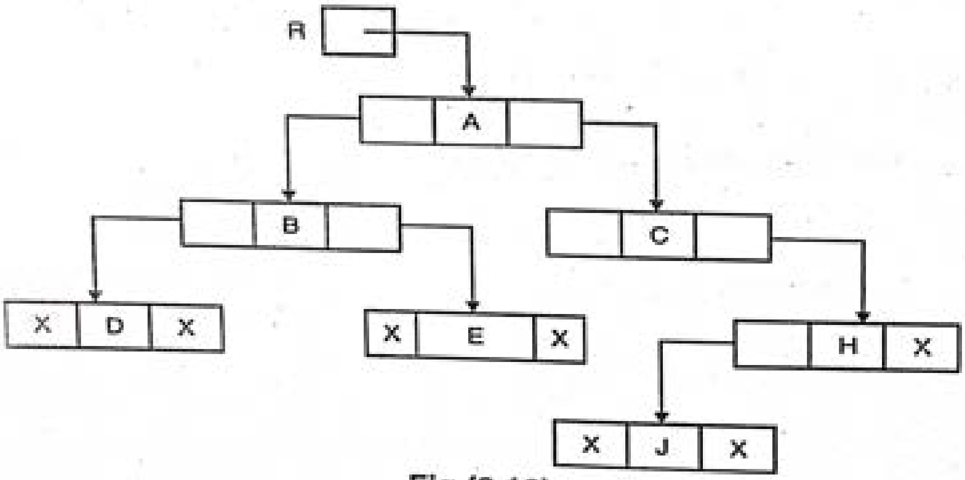


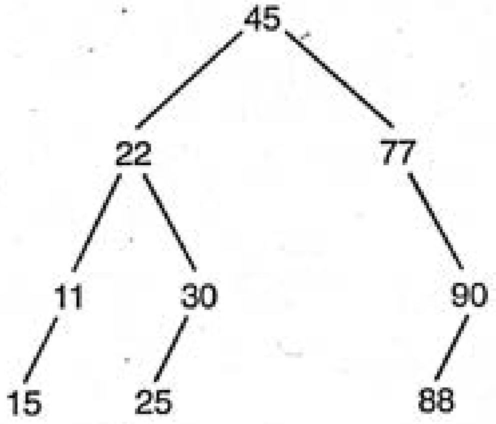
Fig.(2.16)

ROOT
5

	INFO	LEFT	RIGHT
1			
2	C	0	4
3	E	0	0
4	H	6	0
5	A	10	2
6	J	0	0
7	D	0	0
8			
9			
10	B	7	3

Sequential Representation

Suppose T is complete binary tree then only single linear array TREE is used as follows:



1	45
2	22
3	77
4	11
5	30
6	
7	90
8	15
9	
10	25
11	
12	
13	
14	88

Fig. (2.17)

1) The root R is stored in TREE [0]

1) If node N occupies TREE [K] then left child is stored in TREE [2 * K] and its right child is stored in TREE [2 * K + 1]

2) If TREE [1] = NULL then it is empty.

2.10 STACK AND QUEUE DATA STRUCTURE

A stack is a list of elements in which an element may be inserted/deleted only at one end, called the top of stack. Elements are removed from a stack in a reverse order of that in, which they were inserted into the stack. So it is generally referred as

LIFO (Last In First Out structure).

Two basic operations associated with stack are as follows

1) Push – To insert an element into a stack.

2) Pop – To delete an element from a stack.

Best example of stack is stack of dishes.

Example: Suppose A is empty stack. Following 5 elements are pushed onto the stack.

1) Pentium
2) 8085 microprocessor
3) 8051 microcontroller
4) 80486
5) 8255 peripherals

Stack is represented in following diagram

Stack A

TOP →	5	8255 peripherals
	4	80486
	3	8051 microcontroller
	2	8085 microprocessor
	1	Pentium

1	2	3	4	5	6 - - -	- - - N
Pentium	8085 microprocessor	8051 microcontroller	80486	8255 peripherals		

Top of Stack

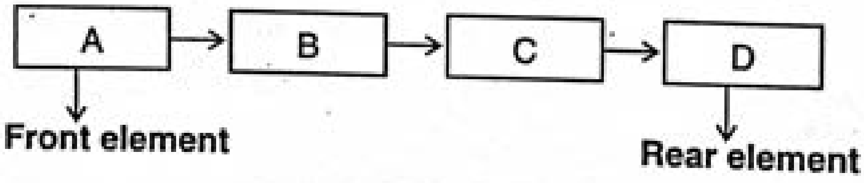
In this case 4th element can't be deleted before 5th element. In the same way for other elements also. The elements may be popped from the stack only in reverse order of the in which they were pushed onto the stack.

Queue

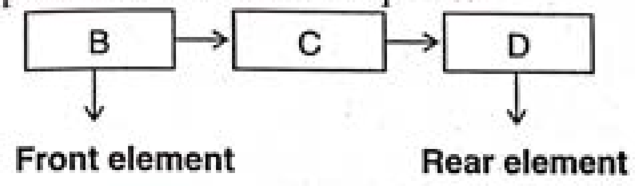
A queue is a linear list of elements in which deletions can take place only at one end called the front and insertions can take place only at other end called the rear. **Queues are also called FIFO (First In First Out lists)** i.e. first element in a queue will be first element out of the queue. Queues abound in every day life. E.g. queue waiting for a bus.

e.g. fig. shows a schematic diagram of a queue.

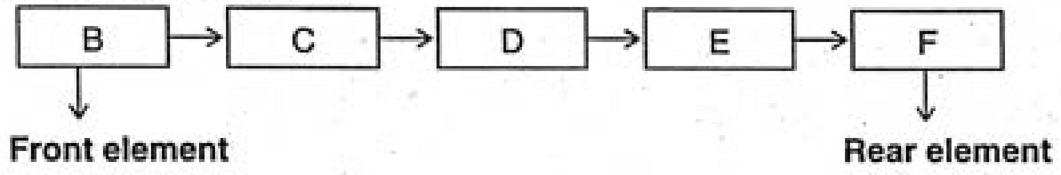
1)



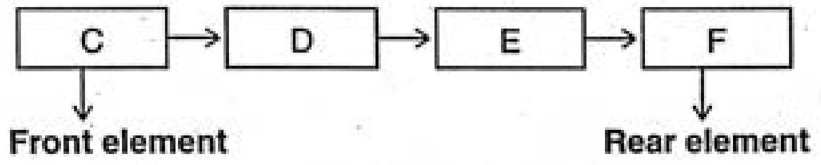
2) Suppose A is deleted from queue:


B
C
D
Front element
Rear element

3) E & F elements are added:


B
C
D
E
F
Front element
Rear element

4) B is deleted:


C
D
E
F
Front element
Rear element

QUESTIONS

1) Select the correct alternative and rewrite the following:

a) Maximum number of nodes of symmetric binary tree with depth of 5 is ____.

i) 5, ii) 25, iii) 31, iv) 32 **(Mar. 2002)**

b) To find the location of record with a given key value is called ____.

i) Searching, ii) Sorting, iii) Finding, iv) Merging **(Mar. 2005)**

c) _______ is very useful in situation when data is to be stored and retrieved in reverse order. **(Mar. 2019)**

i) Stack, ii) Queue, iii) Linked List, iv) Tree

d) The number of comparisons required for bubble sorting of an array of 'n' elements is ____. **(Specimen Paper)**

i) $n(n-1)/2$, ii) $n/2$, iii) $\log_2 n$, iv) $\log_{10} n$

e) ____ is the operation of rearranging the elements of an array either in increasing or decreasing order.

i) Sorting, ii) Searching, iii) DMS, iv) DBMS **(Apr 90)**

f) Accessing each element of an array only once is called ______. (Mar.2003,11)

i) Traversing, ii) Searching, iii) Locating, iv) Selecting

g) The most efficient search algorithm is______. (Mar.2004)

i) Binary Search ii) Reverse Search iii) Linear Search, iv) Pointer Search

h) Finding the location of record with a given key value is known as (Mar.2005)

a) Traversing b) Searching c) Sorting d) Inserting

i) Maximum number of nodes of symmetric binary tree with depth of 7 is --------

a) 125 b) 127 c) 128 d) 124 (Mar. 08)

j) Stored list is essential in ——— process of an array. (Mar.2010)

i) Linear Search ii) Binary Search iii) Traversing, iv) Insertion

k) Record contains ——— data. (Mar.2009)

i) Homogeneous ii) Non-homogeneous iii) same iv) none of these

l) ——— data structure does not require contiguous memory allocation.

a) array b) string c) pointer array d) Linked list (Mar.2015)

m) A record is collection of ——— (Mar.2017)

a) files b) arrays c) fields d) maps

2. Define the terms: a) Field b)Record c)File (March 2009)

3. Explain the following data structures with suitable diagram.

a) Linear array, b) Linked list, c) Tree (Mar.2003,11)

4. What is linked list? Explain it with suitable example.

5. What are types of binary tree? Explain it with suitable example.

6. What is Binary Tree? With a suitable example, explain the terminology. Describe family relationship between the elements of a binary tree. (Mar.05,11)

7. What is a Binary Tree? With a suitable example, show the relationship between total number of nodes and depth of a binary tree. (Mar. 06)

8. How binary trees are represented in memory? Explain it.

9. Write the difference between linear search and binary search. (Mar. 2017)

10. Explain any four data structure operations.

11. Explain data structure array.

12. What are "records"? Explain how records are represented in memory using arrays? **(Mar.2004)**
13. Explain the algorithm for inserting an element from array. **(Mar.2009,2016)**
14. What is a record? How it differs from a linear array? **(Mar. 02, 05, 07, 08)**
15. What is searching? Explain binary search algorithm.
16. Explain linear search algorithm with a suitable example. **(Mar.2003)**
17. What do you understand by the term "Searching"? Which are the searching algorithms? Explain the linear searching algorithm. **(Mar.2004,07)**
18. How Linked list are represented in memory? **(Mar. 2017)**
19. What is meant by Linked list? With suitable examples show the representation of linked list in memory. **(Mar.2003,2004,2019)**
20. What is linked list? Show a linked list with suitable example having six nodes with a properly labeled diagram. **(Mar. 03, Mar. 07, Mar. 08)**
21. With a suitable example and diagram show a link list with information elements and the link fields from start to null pointer. **(March 2005)**
22. With a suitable example, show labeled diagram for link between two nodes having the information part and next pointer field. **(March 2006)**
23. Explain the advantages of binary search algorithm with a suitable example. State any two disadvantages or limitations of binary search. **(Mar. 07)**
24. Explain the bubble sort algorithm with suitable example.**(Mar.02,05,Mar.08,20)**
25. What is sorting? Explain bubble sorting algorithm. **(Mar.2011)**
26. What is binary tree? Draw the tree structure for the following expression:

    $E = (a-b)/[(c^* d) + e]$ **(Mar. 2002)**
27. Draw the tree structure for the following expression:

    $[a + (b - c)] * [(d - e) / (f + g - h)]$ **(Specimen Paper)**

25. What is binary tree? Draw the tree structure for the following expression:

    $E = (a+b)/[(c^* d) - e]$ **(Mar. 2004)**

26 .With suitable example and labelled diagram show the representation of binary tree in memory. **(March 2003)**

27.Draw the tree-diagram which corresponds to the following algebraic expression:

$E = (2X + Y) / (5A - B)^3$ **(Mar. 06)**

28.What is Binary Tree? Draw the binary tree structure for the following expression:

$[(a + b) * c] / [a * (b - c) + a]$ **(Mar. 08)**

29.What is Binary Tree? Draw tree structure for the following expression:

$E = (a - b) / ((c*d) + e)$ **(March 2007)**

30. Show the representation of records in memory considering suitable example of three records and three fields. **(Mar.03)**

31. Explain with flowcharts the following control structures. **(Mar.2009,2017)**

a) Sequential b) Selection logic c)Iteration logic

32. With suitable example show representation of Binary tree in a memory. **(Mar.2009)**

33. Explain bubble sort algorithm with suitable example. **(Mar.2017)**

34. Explain Binary Search algorithm with a suitable example. **(Mar.2019)**

35. What is Binary Tree? Draw the Tree diagram for the expression.

$B = (3R/5T)^2 - (R + Q^3)$ **(Mar.2019)**

36. What are the functions of Memory Management? State any two types of continuous Real Memory Management System. **(Mar.2019)**

37. Explain internal and external fragmentation in memory management of operating system. **(Mar.2020)**

38. Define linked list. Draw and explain labelled diagram of linked list with 5 nodes. **(Mar.2020)**

Answers Q.(1)

(a) 31 (b)searching (c)stack (d)n (n-1)/2 (e)Sorting (f)Traversing (g)Binary search (h) Sorting (i) 127 (j) Binary Search (k) Non-homogeneous (l) Linked list (m) fields

□□□

3

C++ PROGRAMMING

INTRODUCTION

Last year we have seen C++ without its powerful object oriented concepts. In this chapter we will now see the concept behind **object oriented programming,** C++ as OOP, and some of the striking features of OOP.When programmer is given a complex problem, he follows different programming styles to tackle it.

Some of them are listed below.

3.1 PROGRAMMING APPROACHES

- **Unstructured programming.**

The main program directly operates on global data. Here "main program" stands for a sequence of commands or *statements*, which modify data, which is *global* throughout the whole program. These programming techniques generate tremendous problems once the program gets sufficiently large.

- **Procedural programming**

The main program coordinates calls to procedures and hand over appropriate data as parameters. A single program, which is divided into small pieces, called procedures. Modular programming allows grouping of procedures into modules.

- **Modular programming**

The main program coordinates calls to procedures in separate modules and hand over appropriate data as parameters. Each module can have its own data. This allows each module to manage an internal *state*, which is modified by call to procedures of this module.

- **Object-oriented programming**

Object-oriented programming solves some of the problems just mentioned. In contrast to the other techniques, we now have a web (group) of interacting *objects*, each house-keeping its own state. Objects of the program interact by sending messages to each other. It is most advanced approach and offers many advantages over other approaches just mentioned above.

3.2 IDEA BEHIND OBJECT ORIENTED PROGRAMMING

Object Oriented Programming (OOP) is most modern way of organizing programs. Here the emphasis is given on the way programs are designed; in particular OOP programs are organized around objects, which contain both data & functions that act on that data.

The idea is that when a programmer is given a new problem, the decomposition process that initially takes place is in terms of the objects found in the problem. These objects can be classified into different groupings called **classes,** and these classes can then be analyzed to determine:

The **properties** held by objects of the class,
The **actions** objects of the class are capable of performing,
The **relationships** between classes.

One strong example of this is Biology, where the complexity of life around us has been studied extensively in terms of classifying life forms into various groups and subgroups. Using the object-oriented terms of class and instance, a species can be seen as a class and an individual member of the species as an instance (or object) of the class.

Object-oriented design allows us to think about the actual real-world entities of the problem we are attempting to provide a solution for. Here instances refer to objects &class provides a template for number of objects.

Benefits Of OOP

OOP approach has lot many advantages over other programming approaches. Here are some of them listed below:

1) OOP treats data as a important element in the program development and doesn't allow to be accessible by any user. Thus it protects data from accidental modification from outside world. This is achieved through data hiding.

2) OOP allows us to decompose a complex problem into number of entities called objects and then build data and functions around these entities. Here different objects may communicate with each others through functions.

Thus object orientation gives us the solution of many problems associated with developments. Software complexity can be easily managed.

3) Using encapsulation, it is easier to view the actual data and associated function allowing us to get more details of a model. Encapsulation is collection of data and functions together.

4) Inheritance, provides you software reusability, code saving and extending use of existing classes.

5) Through polymorphism, same thing can be implemented in slightly different manner. Providing more flexibility in the design.

Features of OOP:

1) Programs are divided into **objects,** and different objects communicate with each other through messages, called **methods.**

2) Emphasis is given **on data** rather than procedure.

3) Data is **hidden** and can't be accessed or altered by external functions.

4) Functions operating on data of an object are **encapsulated** (tied together in single data structure called class.)

5) New data and functions can be easily added whenever necessary.

6) Follows **bottom–up approach** in program design.

C++ as Object Oriented Language

OO programming techniques are the best ways, to develop large, complex software applications and systems. C++ can be used both as an **OOPL** and simply "as **a better** C". C++ programs are fast and efficient, which helped to make C++, an extremely popular programming language. It uses templates, static polymorphism, dynamic polymorphism, generic programming, inheritance; data binding and encapsulation etc.which covers most of the object-oriented concepts.

Also C++ supports a programming technique that allows memory management to be safe and implicit . In combination with the use of templates, constructors, and destructors enables the C++ programmer to use programming and design techniques that are more advanced than what is supported in the languages with which C++ is most often compared.

Applications of Object Oriented Concepts

There are number of areas in which OOPs concepts can be interpreted and applied. Major application lies in the design of GUI (Graphical User Interface).In case of real time business applications,OOP are mostly used since it is capable of handling complex problems and strong features of it.

It's important role lies in design, maintenance, and modeling of complex systems. Recently it is moved into analysis of the system to be designed. It also provides strong automation support in CAD, CAM, CIM systems. Maintains Object-oriented databases. Also used in AI and Expert Systems Used for simulation, monitoring and modeling of Real time systems.

3.3 OBJECT-ORIENTED TERMS AND CONCEPTS

Objects are the physical and conceptual things we find in the universe around us. Hardware, software, documents, human beings, and even concepts are all examples of objects.e.g. Script writer,financier,actor,actress,spot-boy,lyricist,choreographer,dress-designer would all be objects a film director might consider in directing a movie.. Likewise, a software engineer would consider stacks, queues, windows, and check boxes as objects. They are called as runtime entities in object oriented system. They are also called as class type entities. (since belong to a particular class). They are even called as an **instance of the class.** They are used to access members (data and functions) of the class.

- **Classes**

Up till now we have seen built in data types (int. float,etc.) derived data types (arrays functions) those are already known to compiler. Class is said to be user-defined data type; which is defined by the user. Users of object-oriented technology usually think of classes as containing the information necessary to create instances, i.e., the structure and capabilities of an instance is determined by its corresponding class.
Classes serve as **templates** for the creation of objects.

From object oriented point of view, class provides entire set of data and related functions to be made user-defined. Thus programmer can build number of classes according to requirements.

Thus we can say, a class is a pattern, template, or blueprint for a category of structurally identical items. The items created using the class are called instances. Whenever class is used to create objects, process is known as instantiation.

Instantiation has two common meanings:

- as a *verb*, instantiation is the process of creating an instance of a class, and
- as a *noun*, an instantiation is an instance of a class.

(Some people restrict the use of the term **"object"** to **instances of classes.)**

➢ Operations /methods (functions)

A message is a request to an object to invoke (call) one of its methods. A message therefore contains

- the name of the method and
- the arguments of the method.

A **method(function)** is associated with a class. An object invokes a method as a reaction to receipt of a message. Also many objects communicate with each other via methods.

➢ Data binding and Encapsulation:--

It is one of the strongest features of oops supported by C++. The wrapping up of data and its associated functions together under a single unit is called as encapsulation. While providing access to an object only through its messages, while keeping the details private is called **Information hiding**. Information hiding is the process of hiding all the details of an object that do not contribute to its essential characteristics. Typically the structure of an object is hidden as well as the implementation of the methods. The advantage of using this is, the data is hidden from outside world and hence from accidental modification.

➢ Inheritance

. **Inheritance** can be defined as the process whereby one object acquires characteristics from one or more other objects. Some object-oriented systems permit only **single inheritance,** a situation in which a specialization may only acquire characteristics from a single generalization. Many object-oriented systems, however, allow for **multiple inheritance,** a situation in which a specialization may acquire characteristics from two or more corresponding generalizations.

Thus main purpose behind using inheritance is **reusability** of the existing code.

➢ Polymorphism

- Literal meaning - many forms (ability to take more than one form)
- OOP meaning
- Same method (message) but different behavior (function code)

- Object determines which method gets executed

Polymorphism means existing in many forms, that is same entity like function, operator is implemented in different ways. In C++ polymorphism is implemented using virtual functions, function overloading, and operator overloading.

Now let's see all these concepts in detail.

3.4 CLASSES

As we are aware, classes are data structures clubbing together data and associated functions together. Let's see how,

Class declaration:

```
class class_name
    {
        access-specifier:
                    data and functions
        access-specifier:
                    data and functions
                    ...
                    ...
        access-specifier:
                    data and functions

    } object-list;
```

C++ allows the declaration and definition of classes. Instances of classes are called *objects*. In above declaration, **class** is a keyword. Access specifier may be **public, private** and **protected,** which decides scope and visibility of data members and member functions declared in the class definition. Thus class is said to be user – defined data type. So implementation of the same class varies from user to user. Following example illustrates the class definition including its members:

Example1: Class definition including its members:

```
#include<iostream.h>
class ABC
{
            private: int data;
            public:  void getdata( )
                            {
                                    cout<<"Enter number:";
                                    cin>>data;
                                    cout<<"The number entered is:"<<data;
                            }
};
void main( )
{
            ABC obj;
            obj.getdata( );
}
```

Output:

```
Enter number: 10
The number entered is: 10
```

Program Explanation: Here we have written class definition having class name ABC.In this class we have declared one variable as data which is called as data member of the class, while function getdata is accessing this data member, it is called as member function.

Class definition ends by giving semicolon after closing curly brace. Then we write main function (called nonmember function), in which you instantiate a class by creating a object. Thus a class is loaded in memory, in turn giving memory to its respected members (including data members and member functions). This process is known as instantiation or object creation. Then you give call to member function via object.

Scope rules for different members of the class:

A class in C++ has a group of data and associated functions called data members and member functions, which are given, one of the access modifiers, namely public, private and protected.

1) **public** access modifier:

All data members and member functions declared using public access modifier (i.e. those are defined in the public section of the class) are accessible by any function in a program.

2) **private** access modifier:

If you don't specify, any access modifier to the data members or member functions of the class, they are by default private. These are hidden from outside world and they are normally used to implement data hiding concept of object oriented programming.

But only member functions and friends of the class can use them, in which are declared.

3) **protected** access modifier:--
These can only be used by member functions and friend of a class as well as derived class member functions. They are similar to private members since they can't be accessed directly by non-member functions, but can be used by derived ones. Therefore this access modifier is more restrictive than public but less restrictive than private. Following program illustrates the use of access modifiers.

Example 2: A Program using access modifiers to display object details.

```
#include<iostream.h>
class ABC
{
        private:int number;
        protected:int data;
        public: void getdata( )
                {
                        cout<<"Enter first number:";
                        cin>>number;
                        cout<<"Enter second number:";
                        cin>>data;

                  }
                  void putdata( )
                  {
                          cout<<"The first number entered is"<<number;
                          cout<<"The second number entered is"<<data;


                  }
};
void main( )
{
        ABC obj;
        //obj.number=10;     //can't access private members.
        obj.getdata( );
        obj.putdata( );
}
```

Output:

```
                Enter first number:10
                Enter second number:20
                The first number entered is10
                The second number entered is 20
```

Program Explanation: In above program, data is protected member, while number is a private data member. Both of these can't be accessed from within non-member function i.e. main function. But getdata( ) and putdata( ) can access them since they are member functions of the class. Thus from main function you can access these private and protected members, not directly, but through public member functions, here getdata( ) and putdata( ).

A class itself may be defined at any one of three possible scopes:

1. At the global scope. This leads to a global class, whereby it can be referred to by all other scopes. The great majority of C++ classes (including all the examples presented so far in this chapter) are defined at the global scope.
2. At the class scope of another class. This leads to a **nested class**, where another class contains a class.
3. At the local scope of a block or function. This leads to a local class, where the class is completely contained by a block or function.

3.5 OBJECT IN C++

Objects are autonomous entities of the class; they are instances of the class and belong to a class. Traditionally, code and data have been kept apart but in *o-o* (object-oriented) programming, code and data are merged into a single indivisible thing ‘ an object’. Thus methods are normally invoked using objects.

Example 3: Following program illustrates invoking class members (data and function) via instance (objects) -

```
#include <iostream.h>
class ABC
{
   public:   int i;
             void display( )
            {
                    cout<<”Invoking member function through object”;
             }
};
void main( )
{
          ABC obj;           //instance of a class.
          obj.i=5;           //accessing data member via instance
          cout<<obj.i;
          obj.display( );    //accessing  member function via instance
}
```

Output is : 5   Invoking member function through object

➢ Pointers to Object Members

We have seen use of pointers, to point to a variable. Pointers can also be used to point object (members of a class also).

Example 4: Accessing members of a class using pointers (pointers to object members of a class):

```
:#include <iostream.h>
class ABC
{
        public:
                int data;
                void getdata( )
                {       cout<<"Enter data:";
                        cin>>data;
                        cout<<"The number entered is:"<<data;
                }
};
void main( )
{
        ABC obj;
        int *p;
        obj.data=4;
        p = &obj.data;          // pointer to object of a class
        cout << *p;
        ABC *q;
        q = &obj.data ;
        q->getdata( );
}
```

Output is:

```
4
          Enter the number : 10
          The number entered is : 10.
```

Program Explanation: In above program, we have 2 pointers, one is integer pointer and other is class type pointer. Both are used to point to object members.

Thus both data members and member functions can be accessed via pointers also, just by giving them, reference of class to which they belong to.

For invoking member function through pointer, we use **pointer to member function operator (->)**

Thus you can assign the address of a public member of an object to a pointer.

Then access that member by using the pointer.

When accessing members of a class given a pointer to an object, use the arrow (->) operator.

Passing References to Objects

Variables can be passed to the function either by value or by reference. Similarly we can pass a reference to object to a function. When an object is passed as an argument to a function, a copy of that object is made, i.e. function creates its own copy and works on its own copy therefore, any changes made in function don't affect the original object while when you pass an object by reference, no copy of the object is made, its memory address is passed to a function so that called function directly works on the original object which is passed to the function during function call. Therefore any changes made in the function are reflected in the original object.

Example 5: A program illustrating how to pass reference to object as a function argument.

```
#include <iostream.h>
class ABC
{
        public: int data;
                void getdata( )
                {
                        cout<<"Enter number:";
                        cin>>data;
                        cout<<"You entered:"<<data;
                }
                void putdata(ABC &obj)   // passing reference to object
                {
                        obj.data = -obj.data;
                        cout<<"Now number is changed to:"<<obj.data;
                }
};
void main( )
{
        ABC obj1;
        obj1.getdata( );
        obj1.putdata(obj1);
}
```

Output is:

```
Enter number: 10
You entered: 10
Now number is changed to: -10
```

Program Explanation: Here in the putdata( )function we have passed reference to object to modify the data member from within the member function via it's object.

➢ An Array of Objects

Last year we have already discussed the concept of array, but we have restricted this idea only to built in data types. But it can be extended and used with the user-defined data types.

Thus C++ supports arrays of any data type., including class type. So we can call array having class type elements as array of objects.

Syntax : classname obj[dimension];

e.g. **ABC obj[3];**

Here is declared array of 3 objects of class ABC

Example 6: Program using array of objects

```
#include<iostream.h>
class ABC1
{
        private:int number;
        public:void getdata( )
                {
                        cout<<"\nEnter number:";
                        cin>>number;
                        cout<<number;
                }
                void putdata( )
                {
                        cout<<"The number entered is"<<number;
                }
};
void main( )
{
        ABC1 obj1[3];
        ABC1 obj;
        for(int i=0;i<3;i++)
        {
                obj1[i].getdata( );
                obj1[i].putdata( );
        }
}
```

Output is:

```
Enter number: 10
The number entered is10
Enter number: 20
The number entered is20
Enter number: 30
The number entered is30
```

3.6 METHODS/FUNCTIONS

They are used to manipulate the data. There are two ways to define methods:

1.Methods can be defined **in class definitions.**

e.g. int getdata( ){ return number1; }

C++ compilers treat these as inline functions: they try to expand the bodies of the functions where they are called.

2.Methods can be defined **outside the class definitions.**

It is also possible to merely declare a method in a class definition,

int get_data( );

and give the **full definition later**, outside the class: using scope resolution operator.

int ABC:: getdata(){ return number1; }

i.e. getdata is a function from class ABC. Because this is outside the class, we must qualify the function name with the class (ABC::).

Example 7: Following program illustrates the same.

```
#include <iostream.h>
class ABC
{
        public:
                int data;
                void getdata( )
                {       cout<<"Enter number:";
                        cin>>data;
                }
                void putdata( );
};
void ABC::putdata( )
{
        cout<<"Entered number is:"<<data;
}
void main( )
{
        ABC obj;
        obj.getdata( );
        obj.putdata( );
}
```

Output is:

```
Enter number: 10
Entered number is: 10
```

Program Explanation: Here putdata( ) member function is declared outside the class definition, so we specify scope resolution operator and classname to which this function belongs to.

3.7 CLASSES VERSES STRUCTURES

A structure is a class whose members are by default public. (Remember that all of the members of a class are by default private.) Structures are defined using the same syntax as classes, except that the keyword struct is used instead of class. For example,

```
struct Rectangle
{
        Rectangle(int, int);
        void draw(int, int);
        int  x, y;
};                              is equivalent to: --

class Rectangle
{
 public:
        Rectangle( (int, int);
        void draw(int, int);
        int  x, y;
};
```

The struct construct originated in C, where it could only contain data members. It has been retained mainly for backward compatibility reasons. In C, a structure can have an initialiser with syntax similar to that of an array. C++ allows such initializers for structures and classes all of whose data members are public:

EXERCISE

a) **Define class complex with appropriate data and function members.**
b) **Write a program in C++ to display a given complex number.**
c) **Write a program in C++ to accept and display three complex numbers.**

3.8 NESTED CLASSES

We have already seen nesting of loops,i.e. defining loop within a loop. Similarly you can define a class inside other class definition. So this is called nesting of the class. The name of such a nested class is local to the nesting class.

Example 8: Following example illustrates the use of nested class.

```
#include <iostream.h>
class A
{
      private:int adata;
      public: void getadata( )
```

```
                {
                        cout<<"Enter the number:";
                        cin>>adata;
                        cout<<"hi you are in outer class and you
        entered:"<<adata;
                }
                class B
                {
                        private:int bdata;
                        public: void getbdata( )
                                {
                                        cout<<"Enter the number:";
                                        cin>>bdata;
                                        cout<<"hi you are in inner class and you
entered:"<<bdata;
                                        //getadata( );

                                }
                };
                B bobj;                         //valid.
                void display( )
                {
                        bobj.getbdata( );
                }};
void main( )
{
        A aobj;
        //B bobj;                       //invalid.
        aobj.getadata( );
        aobj.display( );
}
```

Output is:

```
Enter the number:10
 hi you are in outer class and you eneterd:10
Enter the number:20
 hi you are in inner class and you eneterd:20
```

Program Explanation: Here class A is outer class defined, in the public section of which you define nested class B.From within class B,it is not possible to invoke an outer class member function. Also inner class is instantiated in the body of outer class. You can very well invoke inner class member function through its instances from within member function of an outer class. Here we have invoked getbdata( ) function through its instance from within member function display( ) of outer class A.

3.9 CONSTRUCTORS

A **constructor** is an operation that has the ability to change the state of an object. The initialization of object takes place using special function called constructor. It is called when you instantiate an object. The constructor is a function whose name is the same as the class, with **no return type** (not even void).

Constructors are methods that are used to **initialize an** object at its definition time. Constructors have the **same name of the class** (thus they are identified to be constructors). As other methods, they can take arguments. So they can be overloaded. Constructors are implicitly called when we define objects of their classes. There are two types of constructors.

Syntax:

1)default constructor: If you do not define any constructor in your class then system provides it. It has no body but you can define it as well. It is also known as no **argument** constructor.

```
classname( )
{
      //body of default constructor.
 }
```

2)parameterized constructor: this type of constructor can take arguments so it is called **parameterized** constructor.

```
classname(parameter list)
{
    //body of parameterized constructor.
}
```

Example 9: Program using different types of constructor.

```
#include <iostream.h>
 class ABC
 {
        int data;
        public:          ABC( )    // default constructor
              {
                         cout<<"This is default constructor.";
              }
              ABC(int a)      // single argument constructor
              {
                         cout<<"This is parameterised constructor.";
                         data=a;
                         cout<<"value of data is:"<<data;
              }
};
```

```
void main( )
{
        ABC obj;      //default constructor is invoked.
        ABC obj1(5);
}
```

```
Output is : This is default constructor.
            This is parameterized constructor.
            value of data is : 5
```

Program Explanation: Here we have defined class ABC, so constructor's name is also same as classname. We have defined 2 constructors, The first one is default constructor, taking no parameter and is invoked when class object is created(i.e. by statement ABC obj; Here we have provided body for the default constructor.

Now in the second line class object is created with parameter 5,so parameterized constructor is called in the body of which we initialize the data member defined in the class.

Thus with the help of constructors we have fulfilled one of our requirements of implementation of abstract data types: Initialization at definition time.

- **Overloading Constructors**

Like any other function, a constructor can also be overloaded with several functions that have the same name but different type or number of parameters. Remember that the compiler will execute the one that matches at the moment at which a function with that name is called.

In fact, in the cases where we declare a class and we do not specify any constructor the compiler automatically assumes two overloaded constructors ("*default constructor*" and "*copy constructor*"). For example, for the class:

```
class ABC
{
        public:
            int a,b,c;
            void multiply (int n, int m) { a=n; b=m; c=a*b; }
};
```

with no constructors, the compiler automatically assumes that it has the following constructor member functions:

- **Empty constructor**

  It is a constructor with no parameters defined as ***nop*** (empty block of instructions). It does nothing.

```
            ABC::ABC () { };
```

- **Copy constructor**

It is a constructor with only one parameter of its same type that assigns to every nonstatic class member variable of the object a copy of the passed object.

```
        ABC::ABC(const ABC& obj1)
        {
         a=obj1.a;
         b=obj1.b;
         c=obj1.c;
        }
```

Note: It is important to indicate that both default constructors: the *empty constructor* and the *copy constructor* exist only if no other constructor is explicitly declared. In case that any constructor with any number of parameters is declared none of these two default constructors will exist

Example 10: Program using copy constructor.

```
#include <iostream.h>
   class ABC
   {
         private: int data;
        public:  ABC( ){            }
              ABC(int value){       data=value; }
              ABC(ABC &obj)
              {
                    data=obj.data;
                    cout<<"copy constructor invoked:";
              }
    };
    void main( )
    {
      ABC obj1(50);
      ABC obj2;
      ABC obj3(obj1);           //invokes copy constructor.
      obj2=obj1;                //invokes copy constructor.
        }
```

When initializing an instance with another instance (as might happen when declaring an instance and immediately initializing it with the values from another instance) and when passing an instance as a value parameter to a function, the property values of one instance are copied to another instance. Such copying requires a copy constructor. C++ automatically provides one that works as long as the class declaration does not include pointer, class, struct, or array-based data members. In these situations it may be necessary for programmers to declare their own copy constructors

Example 11: Following example illustrates how to overload class constructors

```
#include <iostream.h>
class ABC
{
```

```
          int width, height;
           public:    ABC( );
                     ABC (int,int);
                    int area ( )
                    {
                         return (width*height);
                    }
};
ABC::ABC( )              //default constructor.
{
         width = 5;
         height = 5;
}
ABC::ABC(int a, int b)        //parameterized constructor.
{
         width = a;
         height = b;
}
void main ( )
{
       ABC obj1 (3,4);
       ABC obj2;
       cout << "obj1's area: " << obj1.area( ) << endl;
       cout << "obj2 'sarea: " << obj2.area( ) << endl;
}
```

Output is:

```
obj1's area: 12
obj2's area :  25
```

Program Explanation: Here first constructor is default constructor taking no arguments and when the class is instantiated,(here obj2)the call to this is made ,initialing height and width to be equal to 5.While when obj1 is created since it is called with 2 arguments ,a parameterized constructor is called.

Constructor and operator new:

The arguments of a constructor can be given as parameters when allocating new objects:

```
Date* d1 = new Date( 2000, 2, 23 );
Date* d2 = new Date; // default constructor is called.
```

Exercise: Write a program in C++ to overload constructors. Program should contain two constructors; one without parameter and second with parameters (assume complex class).

3.10 DESTRUCTORS

A mechanism, which automatically "destroys" an object when it gets invalid (for example, because of leaving its scope).. Destructors are used to release any resources allocated by the object's constructor. The usual "resource" being acquired in a constructor (and subsequently released in a destructor) is dynamically allocated memory. You don't call them explicitly (they are called automatically for you), and there's only one destructor for each object. The name of the destructor is the name of the class, preceded by a tilde (~).

Destructors take no arguments. It is even invalid to define one, because destructors are implicitly called at destruction time: You have no chance to specify actual arguments.So overloading of destructor is not possible.

Destruction of objects takes place when the object leaves its scope of definition or is explicitly destroyed. Destructors are declared similar to constructors. Thus, they also use the name prefixed by a tilde (~) of the defining class:

Example 12: Following example illustrates use of constructors and destructor

```
#include <iostream.h>
class ABC
{
        int *width, *height;
        public:         ABC(int,int);           //constructor.
                ~ABC( )     ;                   //destructor.
                int area ( )
                {
                        return (*width * *height);
                }
};
ABC::ABC (int a, int b)
{
        width = new int;
        height = new int;
        *width = a;
        *height = b;
}
ABC::~ABC ( )
{
        delete width;
         delete height;
}
void  main ( )
{
        ABC obj1 (3,4), obj2 (5,6);
        cout << "Area of rectangle is: " << obj1.area() << endl;
        cout << "Area of rectangle is:" << obj2.area() << endl;
}
```

Output is:
```
Area of rectangle is: 12
Area of rectangle is: 30
```

3.11 STATIC MEMBERS

Static members of a class are not specific to an instance of this class. It is common and accessible by all the instances of this class. The class constructor does not initialize static variables. Thus a static function can be considered as a specific global function, which can access the static private/protected variables of the class.

Declaring a member variable within a class as 'static' means that it is shared among all of the class's member functions (and friends), and it has "program" (not auto) lifetime. Its value is not associated with any object of that class type, but rather it exists apart from any such objects. Therefore to invoke it, requires no object creation. Its name has class scope (and thus is subject to access restrictions based on whether it is declared as 'public', 'protected', or 'private'). It has "external" linkage, meaning that its name is visible outside the file in which it is defined.

Access rules for static and non-static functions of a class:

	Non static member variable	**Non static member function**	**Static member variable**	**Static member function**
Non static member function	Yes	Yes	Yes	Yes
Static member function	No	No	Yes	Yes

Static variables are automatically declared to 0,so the explicit initialization in the definition is unnecessary unless you want it to have a non-zero initial value.

Example 13: Program to illustrate properties of static members.

```
#include <iostream.h>
class ABC
{
        private:        static int data;
        public:   static void getdata( )
                {
                        ++data;
                cout<<"count is:"<<data;
        }
        void putdata( )
        {
                ++data;
                cout<<"Now count is:"<<data;
        }
};
int ABC::data=5;
void main( )
{
    ABC obj;
    ABC::getdata( );
    obj.putdata( );
}
```

Output: count is: 6   Now count is: 7

Program Explanation: In above program we have used 2 static members, one is static data member, note that we can initialize this data member to give it nonzero value without requiring constructor(explained after this topic),or even need not be called with the help of object. second static member is a member function getdata( ),which also need not be called via class object since declared to be static. But if we call putdata( ) member function in the similar way,i.e. without instantiation, since being nonstatic compiler will give error message as illegal call of non-static member function. Thus for invoking nonstatic members of the class you need to instantiate the class.

Exercise: Write a program in C++ to count number of objects declared in a program using static function (assume complex class).

3.12 INHERITANCE

One of the most useful features of classes is inheritance. You might want to review the concept. It is possible to declare a class that **inherits the properties** of another class or classes. This means that, with good class design, you can build applications, which are based on proven re-usable code. i.e. main purpose behind inheritance is **code-reusability**.

What you need to do is to create a **subclass/** a ***child*** class (or **derived class,** in C++ terminology) of the original class. This new class **inherits** all the existing messages, and therefore, all the behavior of the original class. The original class is called the **parent class,** or **superclass,** of the new class. A subclass is said to be a **specialization** of its superclass, and the conversely a superclass a **generalization** of its subclasses.

C++ distinguishes two types of inheritance: **public *and* private**: As a default, classes are privately derived from each other. Consequently, we must explicitly tell the compiler to use public inheritance. The type of inheritance influences the access rights to elements of the various super classes

Summary of Inheritance Access Control

		Inheritance Control, (Derived class)		
		Public	**private**	**Protected**
Access Control, (Base Class)	**public**	Public	private	protected
	private	Not accessible	Not accessible	Not accessible
	protected	protected	private	protected

The leftmost column lists possible access rights for elements of classes. It also includes a third type protected. This type is used for elements which should be directly usable in subclasses but which should not be accessible from the outside. Thus, one could say elements of this type are between private and public elements in that they can be used within the class hierarchy rooted by the corresponding class. In general, it is good practice to make data members private. Member functions that must be called from outside the class should be public, and member functions that are only called from within the class should probably be private.]Thus in addition to speeding development time, proper class construction and reuse results in far fewer lines of code, which translates to less bugs and lower maintenance costs.

3.13 DIFFERENT FORMS OF INHERITANCE

C++ supports single and multiple inheritance. Multiple inheritance in C++ is the ability to have two or more base (or parent) classes. The benefits for the user would be to create objects, which would have the characteristics of both its parent or base classes. Thus,

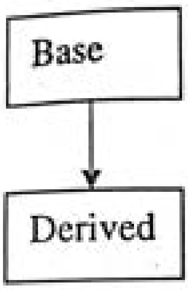
- **Single Inheritance**
  - ➢ Derived class has only one direct base class
  - ➢ Creates "simple" hierarchy of classes - trees
  - ➢ One to one inheritance of members
  - ➢ Specializes a single base class

- **Multiple Inheritance**
  - ➢ Derived class has more than one direct base class
  - ➢ Creates "complex" hierarchy of classes - graphs
  - ➢ Possible multiple inheritance of members
  - ➢ Combines multiple classes

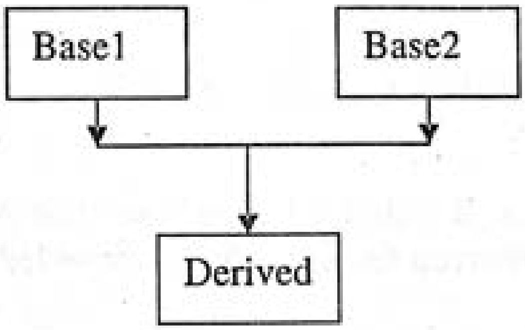
    e.g. `class D : public A, public B { ...};`
  - ➢ Same Inheritance and Access Rules

Multiple Forms of Inheritance

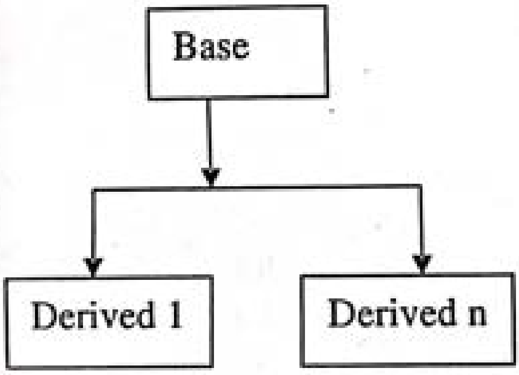
1.Single inheritance


Base
Derived

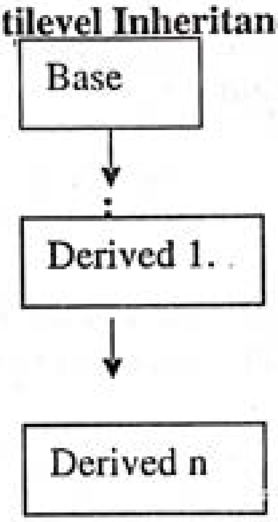
2. Multiple Inheritance


Base1
Base2
Derived

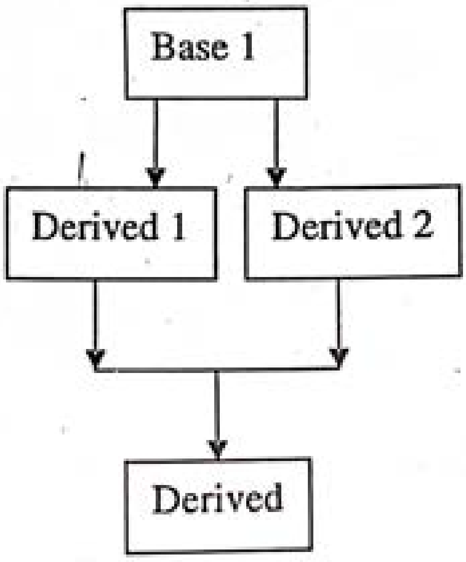
3.Hierarchical Inheritance:


Base
Derived 1
Derived n

4.Multilevel Inheritance


Base
Derived 1.
Derived n

5.Hybrid Inheritance


Base 1
Derived 1
Derived 2
Derived

Syntax for the declaration of derived class

```
class derived_class_name  :  access_specifier base_class_name
{
        // body of class
};
e.g:
class A
```

```
{
//body of class A
};
class B : public A
{
// body of class B
};
```

Here class B is derived from base class A .Here public access specifier defines ,how Class is derived from base class .Now let's see different ways of accessing base class members

Example 14: Base Class Access Control, Public(publicly derived class)

```
#include  <iostream.h>
class base
{
        int i, j;
        public:
                void set(int a, int b) {i=a; j=b;}
                void show( ) {cout << i << " " << j << "\n";}
};
class derived : public base
{
        int k;
        public:
                derived(int x) //derived class constructor.
                {        k=x;    }
        void showk( ) {         cout << k << "\n";        }
};
void main( )
{
        derived ob(3);
        ob.set(1, 2);   // access member of base
        ob.show( );     // access member of base
        ob.showk( );  // uses member of derived class
}
```

Output is:

```
   1       2
   3
```

- When the access specifier for a base class is public, all public members of the base become public members of the derived class; and

- All protected members of the base become protected members of the derived class.
- The base class' private elements remain private to the base and are not accessible by members of the derived class.

// This program won't compile.

Base Class Access Control, private(privately derived)

```
#include  <iostream.h>
class base
{
        int i, j;
        public:
                void set(int a, int b) {        i=a;        j=b;      }
                void show( ) {      cout << i << " " << j << "\n";        }
};
// Public elements of base are private in derived.
class derived : private base
 {
        int k;
        public:   derived(int x) {    k=x;      }
                  void showk( ) {     cout << k << "\n";              }
};
void main( )
{
        derived obj(3);
        obj.set(1, 2);          // error, can't access set( )
        obj.show( );            // error, can't access show( )
}
```

- When the base class is inherited by using the private access-specifier, all public and protected members of the base class become private members of the derived class.
- This means that they are still accessible by members of the derived class but cannot be accessed by parts of your program that are not members of either the base or derived class.

❑ Inheriting Protected Members, public

```
#include <iostream.h>
class base
 {
        protected:    int i, j;     // private to base, but accessible by derived
        public:                   void set(int a, int b) {        i=a; j=b;          }
```

```
                    void show( ) {          cout << i << " " << j << "\n";          }
 };
 class derived : public base
 {
        int k;
        public:         // derived may access base's i and j
                void setk( ) {  k=i*j; }
                void showk( ) {        cout << k << "\n";          }
};
void main( )
{
        derived obj;
        obj.set(2, 3);  // OK, known to derived
        obj.show( );    // OK, know to derived
        obj.setk( );
        obj.showk( );
}
```

Output is:

```
2       3
6
```

- When a class member is declared as protected, that member is not accessible by other, non-member elements of the program just like private members.
- If the base class is inherited as public, then the base class' protected members become protected members of the derived class and are, therefore, accessible by the derived class.

❑ Inheriting Protected Members, protected :--

```
#include <iostream.h>
class base
{
        protected:          int i, j;      // private to base, but accessible by derived
        public:            void setij(int a, int b) {       i=a; j=b;        }
                           void showij( ) {    cout << i << " " << j << "\n";       }
};
// Inherit base as protected.
class derived : protected base
{
        int k;
        public:                 // derived may access base's i and j and setij( ).
                void setk( )
               {
                        setij(10, 12);
                        k = i*j;
```

```
            }

            void showall( )
           {
                    cout << k << " ";
                    showij( );
            }
};
void main( )
{
      derived obj;
      obj.setij(2, 3);       // illegal, setij( ) is protected member of derived
      obj.setk( );           // OK, public member of derived
      obj.showall( );        // OK, public member of derived
      obj.showij( );         // illegal, showij( ) is protected member of derived
}
```

- When the base class is inherited as protected, all public and protected members of the base class become protected members of the derived class.

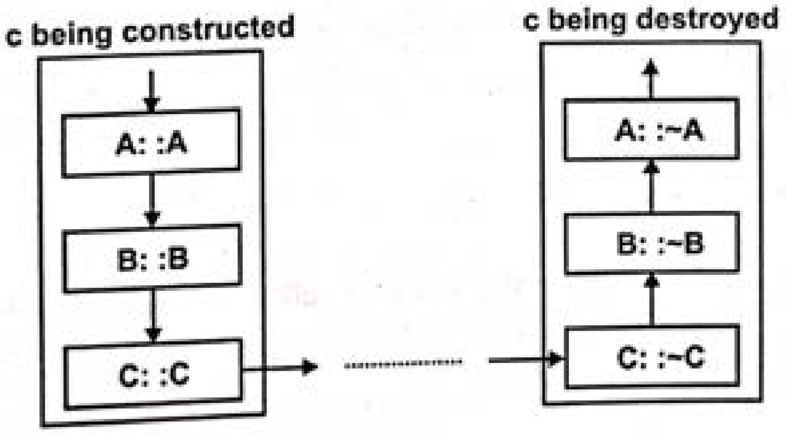
3.14 CONSTRUCTORS AND DESTRUCTOR'S IN DERIVED CLASSES

A derived class may have constructors and a destructor. Since a derived class may provide data members on top of those of its base class, the role of the constructor and destructor is to, respectively, initialize and destroy these additional members.

When an object of a derived class is created, the base class constructor is applied to it first, followed by the derived class constructor. When the object is destroyed, the destructor of the derived class is applied first, followed by the base class destructor. In other words, constructors are applied in order of derivation and destructors are applied in the reverse order. For example, consider a class C derived from B, which is in turn derived from A.

```
class A              { /* ... */ }
class B : public A { /* ... */ }
class C : public B { /* ... */ }
```

Derived class object construction and destruction order.



The constructor of a derived class whose base class constructor requires arguments should specify these in the definition of its constructor. To do this, the derived class constructor explicitly invokes the base class constructor in its member initialization list.

e.g., the XYZ constructor passes its argument to the ABC constructor in this way:

```
                    XYZ::XYZ(const        int         max)        :          ABC(max)
                    { /* ... */ }
```

In general, all that a derived class constructor requires is an object from the base class.

Note: The order in which the base class constructors are invoked is the same as the order in which they are specified in the derived class header (not the order in which they appear in the derived class constructor's member initialization list) and the destructors are applied in the reverse order.

☐ Order of execution of constructors and destructors in inheritance

```
#include <iostream.h>
class base
{
        public:  base( )      {     cout << "Constructing base\n";       }
                 ~base( )     {     cout << "Destructing base\n";        }
};
class derived1 : public base
{
        public:  derived1( )  {     cout << "Constructing derived1\n";   }
                 ~derived1( ) {     cout << "Destructing derived1\n";    }
};
class derived2 : public derived1
{
        public:  derived2( )  {     cout << "Constructing derived2\n";   }
                 ~derived2( ) {     cout << "Destructing derived2\n";    }
};
void main( )
{       derived2 obj;      // construct and destruct obj
}
```

Output:

```
Constructing base
Constructing derived1
Constructing derived2
Destructing derived2
Destructing derived1
Destructing base
```

- When an object of a derived class is created, if the base class contains a constructor, it will be called first, followed by the derived class' constructor.

- When a derived object is destroyed, its destructor is called first, followed by the base class' destructor, if it exists.
- Put differently, constructor functions are executed in their order of derivation.
- Destructor functions are executed in reverse order of derivation.
- In case of multiple inheritance, the general rule applies:
  Constructors are called in order of derivation, destructors in reverse order.

3.15 MULTIPLE INHERITANCE IN C++

You can easily derive from more than one class by specifying the superclasses in a comma-separated list:

· **public** - visible to the world and therefore usable by any other object
· **private** - visible solely within one object i.e. local to an object
· **protected** - visible to an object, its friends and any derived classes

In general, a derived class may have any number of base classes, all of which must be distinct:

☐ Inheriting Multiple Base Classes: --

```
#include <iostream.h>
class base1
{
        protected:      int x;
        public:                   void showx( ) {         cout << x << "\n";       }
};
class base2
{
        protected:      int y;
        public:                   void showy( ) {         cout << y << "\n";       }
};
// Inherit multiple base classes.
class derived : public base1, public base2
{
        public:                   void set(int i, int j) {  x=i; y=j;        }
};
void main( )
{
        derived obj;
        obj.set(10, 20);               // provided by derived
        obj.showx( );       // from base 1
        obj.showy( );       // from base 2
}
```

Output:

```
10
20
```

- To inherit more than one base class, use a common-separated list.
- Be sure to use an access specifier for each base inherited.

Ambiguity in multiple inheritance

Ambiguity in multiple inheritance may also arise when 2 base classes have functions with the same name, while a class derived from both, has no function with this name. So how objects of the derived class can access the correct base class function? Since the function name, alone is not sufficient, since compiler can't figure out which of the 2 functions are to be executed.... This problem is resolved using scope resolution operator to specify the class in which function to be invoked lies.

```
include <iostream.h>
class base1
{
            public:  void show( ) {     cout << "class base1";          }
};
 class base2
 {
            public:   void show( ) {     cout <<"class base2"          }
 };
class derived : public base1, public base2
{
};
void main( )
{
            derived obj;
            obj.show( );                    //ambiguous to the compile
            obj.base2::show( );             //function from base2 class.
}
```

- **Passing Parameters to Base Class Constructors**

Another question arises when dealing with multiple inheritance: how does one pass parameters to the constructor function of a base class, here it is :---

```
#include<iostream.h>
class Base
{
protected:   int j;
public:
        Base(int y)
        {
                  j = y;
                  cout << "Base class constructor called.\n";
```

```
                }
                ~Base( )
                {
                        cout << "Base class destructor called.\n";
                }
};
class Derived : public Base
{
        int k;
public:
                // Derived uses x; y is passed along to Base
                 Derived(int x, int y): Base(y)
          {
                        k = x;
                        cout << "Derived class constructor called.\n";
                }
          ~Derived( )
                {
                        cout << "Derived class destructor called.\n";
            }
};
void main( )
{
 Derived obj1(5,6);
}
```

The output from this program is the same as in the previous example:

```
        Base class constructor called.
        Derived class constructor called.
        Derived class destructor called.
        Base class destructor called.
```

Program Explanation: In this program, however, some values were assigned. After both constructors are called, x has the value of 5 and y has the value of 6.
You should notice that in the class declaration, the base class, Base, requires the parameter int y. Since two parameters are used in Derived (), you ensure that the base class is passed the necessary parameters by using the declaration (assuming that parameter2 is the base class argument):

```
        derived_class(type parameter1, type parameter2, . . .):
        base_class(parameter2)
        {
          // body of derived constructor
        }
```

Granting Access

There are two ways to change the access level of inherited members of a derived class. The first method is a recent development in the world of C++, and it has become the new standard. This new standard is the using declaration. The second way to change an access level is to use **access declarations.**.

The following base and derived class declarations show how access declarations work:

```
class Base
{
        public:    int a;
};
class Derived : private Base
{
        public:    Base :: a;
};
```

Even though, by default, a is inherited by Derived as private, the access declaration

Base :: a ;   changes a's level of access to public.

3.16 USE OF VIRTUAL IN C++ (Dynamic polymorphism)

Virtual Base Classes

Think about the following situation: a class named Base is declared. Then, two classes are derived from that original: Derived1 and Derived2. Next, yet another new class is declared -- a derivative of both Derived1 and Derived2 -- Derived3. Looking at this situation, if a variable from the original class, Base, is called in association with Derived3, what will happen?

There are essentially two copies of Base in Derived3: the one inherited from Derived1, and the one inherited from Derived2. If Derived3 tries to reference a variable in Base, an error will occur. To prevent an extra copy of Base from being created during the declaration of Derived3, we use *virtual base classes.*

The following class declarations clarify the situation described above. Notice the use of the keyword **virtual** in the declaration of the derived classes:

```
class Base
{
        public:
           int a;
};
class Derived1: virtual public Base
{
        public:    int b;
};
```

```
class Derived2: virtual public Base
{
        public:    int c;
};
class Derived3: public Derived1, public Derived2
{
        public:    int d;
};
```

There is only one difference between a virtual class and a "normal" class. If a virtual base class is inherited more than once by the same class, only one copy of it will be present. If a "normal" class is inherited more than once by the same class, multiple copies of it will be present in the derived class. Thus virtual base class prevents inheriting multiple copies of same base class members

3.17 INHERITANCE CONFLICTS

- Member Conflicts
  - Name conflicts can occur - same member name from more than one base class
  - Derived class can overshadow base class members name
  - Use scope resolution operator to resolve conflicts
- Multiple Inheritance Conflicts
  - Derived class may combine more than one copy of a member
  - Base class may combine more than one copy of a member

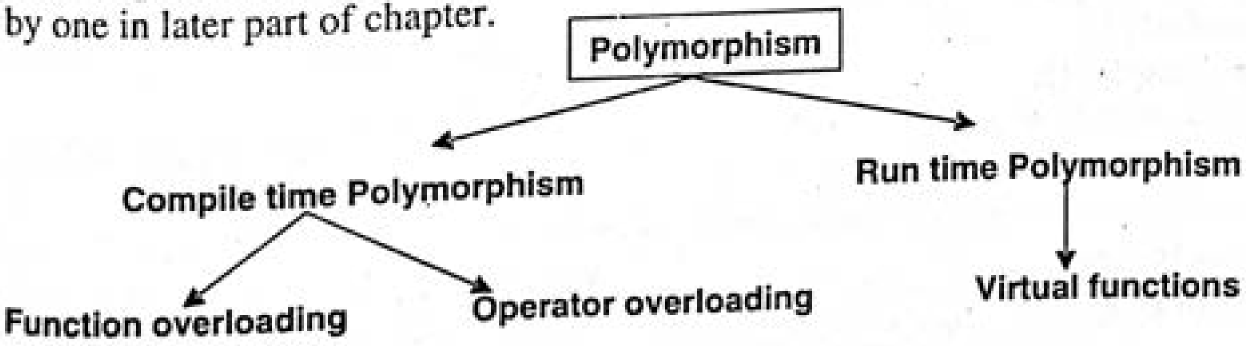
When to use virtual members

Make a member 'virtual' if a derived class extends the functionality of a member of the base class, and this extended functionality has to be accessible. Make a member 'virtual' ---1.inside other member functions of the base class an object of the derived class.

2.When using pointers that can point to an object of the derived class. Virtual function supports runtime polymorphism since ambiguity is resolved at the run time ,so it is also called as **RTTI**(Run Time Type Identification)

Polymorphism:

Before going for polymorphism, user should know types of polymorphism.The types are as given in diagram. We are going in details of that one by one in later part of chapter.



- **Virtual functions:**

Keyword virtual only appears in the base class, all classes that derive this class will have the corresponding method automatically declared virtual by the system. The structure of the virtual function in the base class and each of the derived classes is identical. The return type and the number and types of the parameters must be identical for all since a single statement can be used to call any of them.
If the keyword virtual is used, the system will use late binding, which is done at run time, but if the keyword is not included, early binding will be used. In early binding which function to be executed is decided at compile time so it will execute the function in base class only. In late binding that is decided at run time so it will execute appropriate function from derived class. In late binding it decides function according to contents of pointer while in early it decides function execution according to type of pointer.

Example 15: Following example illustrates the use of virtual function.

```
#include<iostream.h>
class Base
{
public:     void display(){    cout<<"In Base";     }
};
class Derived1: public Base
{
public:   void display(){     cout<<"In Derived1";       }
};
class Derived2: public Base
{
public:   void display(){     cout<<"In Derived2";       }
};
void main( )
{
        Derived1 obj1;
        Derived2 obj2;
        Base *ptr;
        ptr=&obj1;
        ptr->display();
        ptr=&obj2;
        ptr->display();

};
```

```
Output is :-- In Base
              In Base
```

Why so????? Here, compiler ignores the contents of pointer and chooses the member function matching with the type of pointer. (and since pointer is of base type), Base class function is called.
Thus it can't solve the problem of accessing objects of different classes using same statement.

The solution is to use virtual function.

```
#include<iostream.h>
class Base
{
public:     virtual void display(){     cout<<"In Base";     }
};
class Derived1: public Base
{
public:    void display(){      cout<<"In Derived1";        }
};
class Derived2: public Base
{
public:    void display(){      cout<<"In Derived2";        }
};
void main()
{
        Derived1 obj1;
        Derived2 obj2;
        Base *ptr;
        ptr=&obj1;
        ptr->display();
        ptr=&obj2;
        ptr->display();
}
```

Now output is: --In Derived1
In Derived2

Thus, here member function of the respective derived classes is executed. So here same function call(ptr→display();)executes different functions, depending upon the contents of ptr to which it is pointing currently and not on its type(i.e. of base).

NOTE: Pointer rule in C++:--A pointer declared as pointing to a base class can be used to point to an object of a derived class of that base class, but a pointer to a derived class cannot be used to point to an object of the base class or to any of the other derived classes of the base class.

- **Characteristics (Rules) of virtual function:--**

1 The virtual functions must be member of some class.

2.They can't be static members.

3.Object pointers can access them.

4.A virtual function in the base class must be defined, even though it may not be used.

5.Prototype declaration for the virtual function in base and derived class must be same. Since if they are different, it will not be virtual function but compiler will treat it as a function overloading.

6.Virtual constructors are not possible, but virtual destructors are…

7.They can be friend of other class.

8.You cannot use a pointer to a derived class to access base class object.

9.Base class pointer can be incremented or decremented only relative to its base type, and it can't point to next object of the derived class.

10.If virtual function is not defined in the derived one, function of the base class is called.

11.Empty Virtual function defined in the base class is called as pure virtual function

e.g. `virtual void display(void) = 0;`

12.A class declared to have pure virtual function couldn't be instantiated. That means objects of that class can't be created. Such a class is called as abstract base class

13.Virtual functions supports runtime polymorphism (RTTI i.e. Run Time Type Identification.)

Pure virtual function and abstract class

```
class Base
{
        public:     virtual void display()=0;    //pure virtual function.
};
class Derived1: public Base
{
        public:    void display(){       cout<<”In Derived1”;       }
};
class Derived2: public Base
{
        public:    void display(){       cout<<”In Derived2”;       }
        };
void main()
{
        Derived1 obj1;
        Derived2 obj2;
        Base *ptr[2];
        ptr[0]=&obj1;
        ptr[0]-->display();
        ptr[1]=&obj2;
        ptr[1]-->display();
}
```

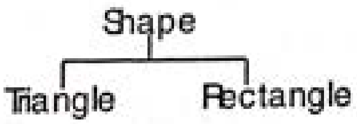
Output is:

```
In Derived1
In Derived2
```

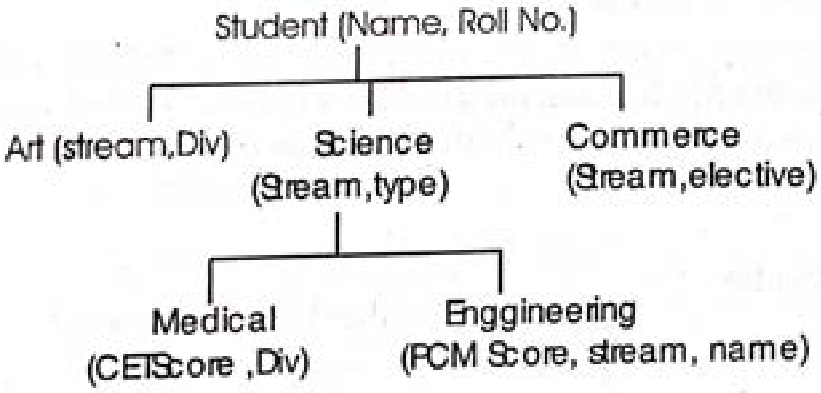
Here virtual function is derived as **virtual void display ()=0;** The equal to sign here has nothing to do with assignment, also value 0 is not assigned to anything…
But the (= 0) syntax is simply, to tell the compiler that a function is pure(i.e. having no body).
But if you remove the body of the virtual function in base class, why can't we remove the function altogether? Since it doesn't work otherwise. So here base class, Base is considered to be as an **abstract class.** Abstract class is often defined as one, that will not be used to create any objects, but exists only as a base class of other classes.i.e. in above program if you create object of the base class to call its function(like Base obj; and obj.display();)it will lead to compiler error---since abstract class can't be instantiated.
The class to be called as abstract, a class must contain atleast one pure virtual function.

EXERCISE

1) Write a program in C++ for the following Inheritance structure:


Shape
Triangle
Rectangle

2) Write a program in C++ to accept and display for the following Inheritance structure:


Student (Name, Roll No.)
Art (stream,Div)
Science (Stream,type)
Commerce (Stream,elective)
Medical (CETScore ,Div)
Enggineering (PCM Score, stream, name)

3) Using virtual function calculate the area of triangle and rectangle.

3.18 FRIENDS IN C++

It is sometimes useful to let the functions of one class have access to the components of another class without making the components public and without the overhead of having to call member functions to get private data, in this case friends are useful. Friend functions or classes have full access to the public, protected, and private members of any class for which they have been declared as friends. The definition of the friend function may or may not be included in the file containing the definitions for the class's member functions

Friends can be either **functions** or other **classes**. A class grants access privileges to its friends. They provide a degree of freedom in the interface design options. The major difference is that a friend function is called like f(x), while a member function is called like x.f().

Derived classes of a friend aren't necessarily friends. If class ABC declares that class DEF is a friend, classes derived from DEF don't have any automatic special access rights to ABC objects. A friend may be a non-member function, a member function of another class, or an entire class. The difference between Friends and derived classes is that derived classes might not be able to access all the members of the base class while Friends can access.

The privileges of friendship aren't transitive. A friend of a friend isn't necessarily a friend. If class ABC declares class XYZ as a friend, and class XYZ declares class LMN as a friend, class LMN doesn't necessarily have any special access rights to ABC objects

The privileges of friendship aren't reciprocal. If class ABC declares that class XYZ is a friend, XYZ objects have special access to ABC objects but ABC objects do not automatically have special access to XYZ objects.

Therefore it is true, even in C++ that "**friendship isn't inherited, transitive, or reciprocal**"

Friend class

One can also declare a whole class to be a friend of another class. This allows all member functions of the friend class to have full access to all the members of the other class.

Syntax: --friend *some-class-name*;

Where this line is placed inside a class declaration and '*some-class-name*' is the name of a class known in the file holding the class declaration The following shows how one class (in this case DEF) can let another class (in this case ABC) be a "**friend**" class

```
#include<iostream.h>
class A
     {
        private: int data;
        public: A( )
```

```
                {
                                data=100;
                }
                 friend class B;
    };
    class B
    {
      public : void friendclass(A aobj)
                {
                          cout<<"value of data is:"<<aobj.data;
                 }
    };
    void main ( )
    {
      B obj;
      A obj1;
      obj.friendclass (obj1);
    }
```

Output is: value of data is: 100

Program Explanation: In this case we have used friend class, here we are accessing private member of class A through its object from within class B's member function.

Friend function:

C++ allows us to make common function friendly to the both classes, thereby allowing function to have access to the private data. Also function need not be member function of the class. Function declaration uses keyword friend, but function definition need not. Also function can be declared to be a friend of any number of classes. So if two or more classes are unrelated (not following inheritance relationship,), friend function acts as a bridge between them.

e.g. **class B;**

```
#include<iostream.h>
 class B;
 class A
 {
         private:   int data;
         public:
                         A() {      data=100;          }
                  friend int  friendfunction(A,B);
 };

 class B
```

```
{
        private:   int data;
        public:    B() {   data=50;   }
        friend int friendfunction (A  aobj,B  bobj);
};

int friendfunction (A  aobj,B  bobj)
        {
                                    return (aobj.data+bobj.data);
        }

void main()
{
            A   obj1;
            B   obj2;
            cout<<friendfunction(obj1,obj2);
 }
```

Output is: 150.

Thus if we want a function to operate on objects of two different classes, (2 objects of different classes, operating on their private data), friend function is used.

Exercise: Writer a program in C++ to swap private data values in two classes using friend function.(Assume single integer data member)

3.19 OPERATOR OVERLOADING

We are already familiar with one of the ways to implement compile time polymorphism by means of function overloading. Other way is operator overloading. Operators are similar to functions in that they take operands (arguments) and return a value. Most of the built-in C++ operators are already overloaded. For example, the + operator can be used to add two integers, two reals, or two addresses. Therefore, it has multiple definitions. But the built-in definitions of the operators are restricted to built-in types. The programmer can provide additional definitions, so that they can also operate on user-defined types. Each additional definition is implemented by a function C++ allows the programmer to define **additional meanings for its predefined operators** by overloading them.

C++ places a number of restrictions on operator overloading. Only the pre-defined set of C++ operators may be overloaded. It is illegal to define a new operator and then overload it. The meaning of an operator must also be preserved. You cannot turn a unary operator into a binary operator or vice versa. and they should follow syntax rules of the original operators. The overloaded operator must have at least one of the operands that is of the user-defined type.

Most computer languages do some overloading of operators as part of the language itself. It is possible, however, in C++ for programmers themselves to re-define the meaning of C++ operators as long as at least one of the operands is an instance of a class

Here are a few of the many examples of operator overloading:

- String1 + String2 are used to concatenate two standard String objects
- Date++ increment a Date object
- Number1* Number2 multiply two Number objects
- arr[i] access an element of an Array object

Defining an Overloaded Operator

Thus to define an additional task to an operator, we must specify what it means in relation to the class to which the operator is applied., so you can include this task in operator's function body. The code itself for an overloaded operator depends on what the programmer wants to happen.

Syntax: returntype classname :: operator oper(parameter list)

```
        {
                function body
        }
```

The word 'operator' is a keyword, followed by the operator symbol and you specify operands as parameters. So this is called as **function operators** or **operator functions.**

The number of arguments (operands) found in an overload declaration depends on whether the operator function is declared as a friend or member function(non static) of the class. (It must be one of the two in order to have access to the members of the class.)

A basic difference between them is that a friend function will have only one argument for unary operators and two for binary operators. While a member function has no arguments for unary operators and only one for binary operators. Arguments can be passed by value or by reference.

Since object used to invoke the member function is passed implicitly and therefore is available to member function, so it requires one argument less than if it is declared as friend function.

Steps for overloading operator:

1.Write the class definition in which operator is to be overloaded, for that you require operator function, where you write the actual code ,you want to execute with the overloaded operator.

2.Declare that operator function as a member function or as a friend function depending on number of arguments that operator is going to take. Also include this in the public section of the class.

3.Now give definition of the operator function, in which you give additional meaning/task to the existing operator.

4.Now invoke overloaded operator,

It is invoked by oper obj; or obj oper

Where obj is user defined data type(may be instance of the class in which operator function is defined) and oper is actual operator to be overloaded.This is to be done if it is unary operator.While if it is binary operator, it can be invoked like,

operator oper (obj1,obj2) if it is a friend function
and obj1.operator oper(obj2) if it is a memberfunction

Important Note:

Overloadable Operators											
Unary:	+	-	*	!	~	&	++	--	()	->	->*
	new	delete									
Binary:	+	-	*	/	%	&	|	^	<<	>>	
	=	+=	-=	/=	%=	&=	|=	^=	<<=	>>=	
	==	!=	<	>	<=	>=	&&	||	[]	()	,

When the compiler encounters an operator it is converted into an appropriate function call.

But note that following operators cannot be overloaded: scope operator (`::'), member object selection operator (`.*'), class object selector operator (`.'), and the arithmetic conditional operator (`?:').and **sizeof** operator.Operators are also not allowed to have default arguments.

Also a strictly unary operator (e.g., ~) cannot be overloaded as binary, nor can a strictly binary operator (e.g., =) be overloaded as unary.

C++ does not support the definition of new operator tokens, because this can lead to ambiguity. Furthermore, the precedence rules for the predefined operators are fixed and cannot be altered. For example, no matter how you overload *, it will always have a higher precedence than +. Operators ++ and -- can be overloaded as prefix as well as postfix.

Example 16: Following example illustrates how to overload unary operator(unary minus):--

```
#include<iostream.h>
class ABC
{
        int no1,no2;
        public:         void getnumbers(int a,int b);
                void display(void);
                void operator -( );
};
void ABC::getnumbers(int a,int b)
{
        no1=a;
        no2=b;
}
void ABC::display(void)
{
        cout<<no1<<endl;
        cout<<no2<<endl;
}
void ABC::operator -( )
{
        no1= -no1;
        no2= -no2;
}
void main( )
{
    ABC obj;
    obj.getnumbers(3,-5);
    cout<<"Before overloading:";
    obj.display( );
    -obj;
    cout<<"After overloading:";
    obj.display( );
}
```

Output : Before overloading: 3 , -5
After overloading: -3, 5

Program Explanation: In this case we are overloading unary operator, since this is unary operator to be overloaded, and defined as member function, this operator function doesn't take any argument. When you define statement (**- obj ;**) , it in turn invokes operator function, which is negating the original number.

Example 17: Following example illustrates how to overload binary operator(+operator to concatenate 2 string objects.)

```
#include<string.h>
#include<iostream.h>
class String
{
        private:
                int size;
                char str[80];
                int no1,no2;
        public:         String()
                {
                        strcpy(str," ");                //default constructor
                }
                String(char s[])
                {
                        strcpy(str,s);
                }
                void display(void)
                {
                        cout<<str<<endl;
                }
                String operator + (String s1)
                {
                        if((strlen(str)+strlen(s1.str))<size)
                        {
                                String temp;
                                strcpy(temp.str,str);
                                strcat(temp.str,s1.str);
                                return temp;
                        }
                        else
                        cout<<"Be cautious!!!!String overflows!!!!";
                }
};
void main( )
{
        String obj1="HI";
          String obj2="HELLO";
        String obj3;
        obj1.display( );
        obj2.display( );
          obj3=obj1+obj2;
        obj3.display( );
}
```

Output is:

```
HI
HELLO
HIHELLO
```

Program Explanation:- Here we have overloaded binary + operator to concatenate 2 strings .Here string objects are user-defined and + operator operates on them, since in the operator function definition, we have concatenated 2 strings, after adding them they are concatenated to each other, i.e second string is appended to the first one .Statement obj3=obj1+obj2 invokes this operator function. Since you want the effect of concatenation in obj3,which is a string object, your operator function must return the object(i.e. class type),therefore return type of this operator function is classtype i.e String.

Dealing with fractions with the operator overloading.:--

This program overloads binary + and / operators to give addition and division of 2 fractions.

```
# include <iostream.h>
class fraction
{
private: float n, d;
public:  fraction()
         {
                 n = 1;
                 d = 1;
         }
        fraction operator +(fraction);          //Operator for adding fractions
        fraction operator /(fraction);          //Operator for dividing same
        void getfr(void);
        void printfr(void);
};
fraction fraction::operator +(fraction f2)
{
        fraction r;
        r.n = n*f2.d+d*f2.n;
        r.d     = d*f2.d;
        return (r);
}
fraction fraction::operator /(fraction f2)
{
        fraction r;
        r.n = n*f2.d;
        r.d = d*f2.n;
        return (r);
}
void fraction::getfr(void)
{
        cout << "\nEnter the numerator  -->";
        cin  >> n;
        cout << "Enter the denominator-->";
        cin  >> d;
}
void fraction::printfr(void)
{
        long double r;
        r = long double(n/d);
        cout << n << "/" << d << " = " << r << " ";
}
void main()
{
        fraction f1, f2, f3, f4;
        f1.getfr();
        f2.getfr();
        f3 = f1+f2;
        f4 = f1/f2;
        cout << "\nFraction 1\t-->";
        f1.printfr();
        cout << "\nFraction 2\t-->";
        f2.printfr();
        cout << "\nSum\t\t-->";
        f3.printfr();
        cout << "\nDivision\t-->";
        f4.printfr();
        cout << endl;
}
```

Output is :

```
Enter the numerator  -->2
Enter the denominator-->5
Enter the numerator  -->4
Enter the denominator-->5
Fraction 1     -->2/5 = 0.4
Fraction 2     -->4/5 = 0.8
Sum           -->30/25 = 1.2
Division       -->10/20 = 0.5
```

EXERCISE

2) **Write a program in C++ to overload increment operation.**
3) **Write a program in C++ to overload +, - , / , and * for addition, subtraction division and multiplication of two complex numbers.(Write a program using both type of funct friend and member function).**

3.20 TYPE CONVERSIONS

When constant and variables of different type are mixed in an expressio compiler applies rules of automatic type conversion.

Assignment operator also causes automatic Type Conversion.

The type of the data on right hand side of an assignment operator is automatically converted to the variable on left.

eg.
```
int a ;
float b = 3.3346 ;
a = b ;
```

It converts float to integer. Here fractional part is truncated. Thus type conversion is automatic for built in data types.

Now consider,

```
obj = obj1 + obj2 ;
```

Now if obj, obj1 & obj2 are of class type objects, compiler will carry out this operation without any objection but it will give error, when one of it is basic data type. Here compiler doesn't support automatic conversion. Thus we have to have conversion routines for: -

a) Conversion from built-in to class type.
b) Conversion from class type to built-in type.
c) Conversion from one class type to another class type.

- **From Basic to user defined**

  The constructor can be used from argument type to constructor's class type

```
string: : string (char * a)
{    len = strlen (a) ;
       p = new char (len + 1);
     strcpy (p, a);
}
```

This constructor builds a string type object from a char * type variable a.

This constructor can be used for conversion from char * type to string type.

eg. str1 = string (name1)

So it converts name1 from char * type to string type and then assigns it to object.

- **From User defined to built in**

  Syntax : operator type name ( )

```
        {
              --- fun
                  statements
        }
```

This fun converts a class type data to type name.

eg.
```
ABC :: operator float ( )
{
        float result = 0;
        for ( int cnt = 0; cnt < size; cnt++)
        result = result * cnt;
        return result;
}
```

This converts a ABC obj to corresponding scalar variable.
Here you can have
float a = obj;
Where obj is of type ABC.

By overloading casting operator in the fun (conversion fun) we can have translation from class to basic type but conversion fun must satisfy conditions as-

1) It must be defined as a class member
2) Must not specify a return value
3) Must not have any arguments

Example 18: Conversion between strings (char array) and string objects.

```
# include <iostream.h>
#include <string.h>
const int size = 80;
class string
{
        private:
                char str [size];
        public:
                string ( )
                {
                        str[0] = '\0';
                }
                string (char s [])              //converts str to string (basic à class)
                {
                        strcpy (str, s);
                }
                void display( )
                {
                        cout << str;
                }
                operator char*( )               // conversion fun
                {       return str;             //converts string to char* (classàbasic)
                }
};
void main( )
{
        string obj1;
        char newstr[] = " \nHI ";
        obj1 = newstr;
        obj1.display( );
        string obj2 = "WELCOME";
        cout << obj2;                   // uses conversion function to convert string to
                                        //   char array before sending to <<operator
}
```

Output is: HI WELCOME

Program Explanation: Here one argument constructor converts a normal string (char array)to object of class string.

This conversion is applied when a string is created as ;

string obj2 = "WELCOME";

Or it will be applied in assignment statement as

obj1 = newstr;

While conversion fun is used to convert from a string class type to a normal string.

eg.
```
operator char * ( )
{
        return str;
}
```

This conversion fun is used by compiler in statement, cout << obj2;
Here obj2 is an argument supplied to the overloaded operator << & as << operator doesn't know anything about user define string type, compiler looks for a way to convert obj2 to type that here << is used to display str(string) & not object .

It finds out conversion fun & uses it to generate a normal string. Which is then sent to calling string:: display ( ) fun.

➢ **Conversion between Objects of different classes;**

Here you can use a one-argument constructor or you can use a conversion fun. Choice depends upon whether you want to put the conversion routine in the class specifier of the source object or of the destination object.

eg. if you say

objA = objB;

↙ ↘

member of class A member of class B.

↙ ↘

(destination) (source)

We can put conversion routine in either destination or source class.

Type Conversion:	Routine in destination	Routine in source
Basic to Basic	(built in conversion functions)	
Basic to Class	Constructor	NA
Class to Basic	NA	Conversion function
Class to Class	Constructor	Conversion function

From Class to Class:

Objects of one class can also be converted to other (object of other class)

Writing conversion routine carries out this conversion.

Example 19: Following program illustrates how to convert object of one class to other.

```
#include<iostream.h>
#include<math.h>
class polar
{
private:
double radius;
double angle;
public:
polar ( )
{
radius = 0.0;
angle = 0.0;
                }
polar (double r, double a)
{
        radius = r;
        angle = a;
}
void display ( )
{
cout <<"Radius="<<radius<<"\t"<< "Angle=" <<angle;
}
double getr ( )
{
return radius;
}
double geta ( )
{
return angle;
}
};
class rec
{
private:
double xcord;
double ycord;
        public:
```

```
rec ( )
                {
                        xcord = 0.0;
                ycord = 0.0;
        }
                rec (double x, double y)
{
xcord = x;
ycord = y;
                }
rec (polar  p)                     //conversion routine 1)
{
                float r = p.getr ( );
        float a = p.geta ( );
        xcord = r * cos (a);
                ycord = r * sin (a);
}
void display ( )
{
        cout <<"X-coordinate=" << xcord <<"\n"<< "Y-coordinate= "
                        << ycord;
                }
        };
        void main ( )
{
        rec reccord;
        polar polcoord (10.0, 0.7855398);
reccord = polcoord;
cout <<"\nPolar coordinates:\n";
polcoord.display (  );
cout<<"\nRectangular Coordinates: \n";
reccord.display (  );
        }
```

Output: - Polar coordinates

```
         Radius=10    Angle=0.785398
         Rectangular coordinates
         X=coordinate=7.07107
         Y=coordinate=7.07107
```

Program Explanation: Here, conversion routine 1) is 1-arg constructors

This function sets the object of which it is a member to the rectangular co-ordinates that corresponds to polar co-ordinates of the object received as an argument.

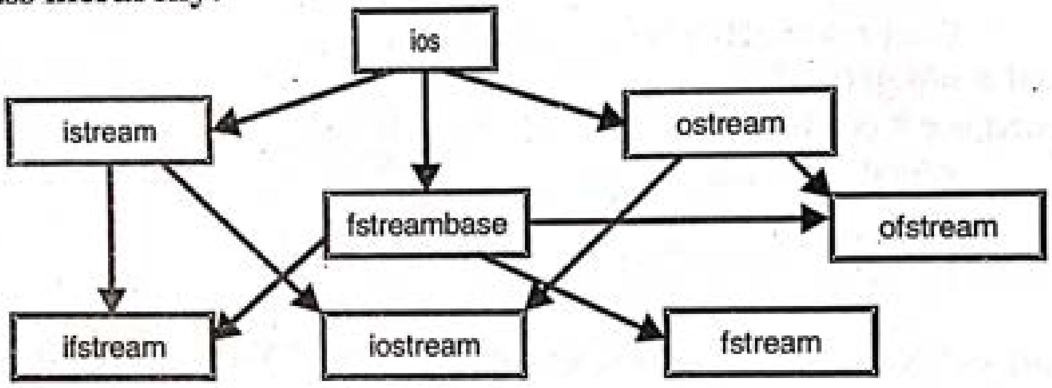
3.21 FILES AND STREAMS

- A stream is a general name given to the flow of data.
- Different streams are used to represent different kinds of data flow.
- Each stream is associated with a particular class, containing member functions and definitions for dealing with that particular flow of data.

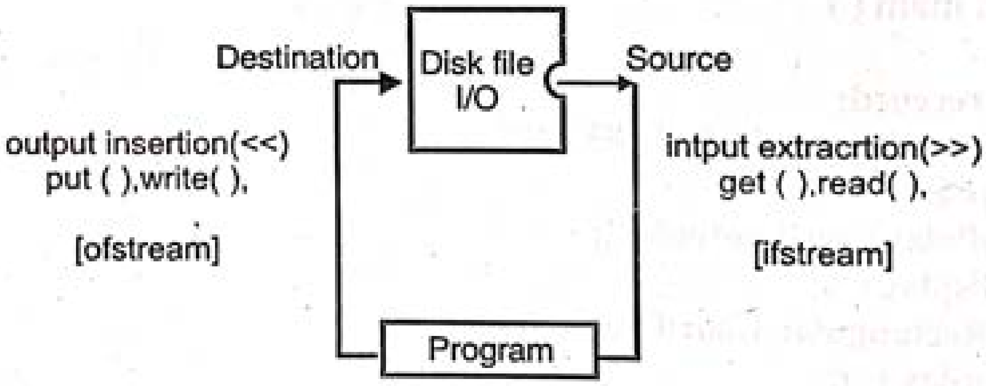
e.g. Extraction operator >> is a member of istream class while insertion operator << is a member ostream class and both of these classes are derived from ios class.

ifstream represents i/p disk files.

Stream class hierarchy:


ios
istream
ostream
fstreambase
ofstream
ifstream
iostream
fstream

Classes used for i/p and o/p to the video display and keyboard are declared in header file, iostream.h while classes used for file I/O are declared in the file fstream.h


Destination
Disk file I/O
Source
output insertion(<<)
put ( ),write( ),
[ofstream]
intput extracrtion(>>)
get ( ),read( ),
[ifstream]
Program

Header Files

ifstream , ofstream and fstream classes are declared in the header file, fstream.h . These files also include iostream.h, so there is no need to include it explicitly, since fstream.h takes call of all stream I/O.

String I/O

```
// o/ps same strings to a file

# include<fstream.h>
void main ( )
{
        ofstream f1("test.doc");
```

```
            f1<<"HI";
            f1<<"Hello";
    }
    }
```

When the program terminates, f1 object of ofstream class, goes out of scope, which in turn calls the destructor, which closes the file. So here we don't need to close the file explicitly. So when you run this program, lines/strings specified, are written to the file and there is no o/p in the screen.

Reading Strings

To read the same file, we must create an object of class ifstream.

```
    #include<fstream.h>
            void main( )
            {
                    const int MAX = 80;
                    char buffer [MAX];
                    ifstream f1("test.txt");
                    while (f1)
                    {
                            f1.getline (buffer, MAX);
    cout<<buffer;
                    }
            }
```

Here the insertion operator doesn't work, but instead we read the text from the file, one line at a time using getline ( ) function.

This function reads the characters until it encounters '\n' character and places the resulting string in the buffer supplied as an argument and contents of the buffer are displayed after each line.

Detecting EOF (End Of file)

An ifstream object (f1 here) has a value that can be tested for various error conditions.If a condition is true, object returns a 0 value, otherwise non-zero. One of these conditions is end of file (EOF).The EOF is a signal sent to the program from h/w, when a read or write operation has reached to the end of the file.

The program checks for the EOF in the while loop so that it can stop reading after the last string. The value returned by a stream pointer is actually a pointer, but the address returned has no significance except to be tested for a 0/non-zero value.

Character I/O

Get ( ) & put ( ) functions, which are members of istream & ostream respectively are used to i/p & o/p single character of a time .

Example 20: File output ,with characters.

```
# include <fstream.h>
# include < string.h>
void main()
{
        char arr[ ]= “HI & Welcome\n”;
        ofstream f1(“Test. txt”);
        for (int  i = 0, i <strlen(arr), i++)
                fl.put (arr[i]);
}
```

File input ,with characters.

```
#include < fstream.h>
void main ()
{
        char ch;
        ifstream f1 (“Test.txt);
        while (f1)
        {
                fl.get (ch);      // read char.
        }
        cout<<ch;                          // display
}
```

This program uses get ( ) function & continues reading it until end of file, is encountered. And each character read from the file is displayed using cout.

3.22 FILES HANDLING IN C++

In real life problems, we need to handle large amount of data. For this we need to use some devices like floppy disk/hard disk to store the data. This data is stored in this device using the concept of files.

A file is a collection of related data stored on the disk.
Programs can be designed to perform read &write operation on these files.

Opening & closing the file :

Syntax: **File-stream-class stream object;**
stream object open (“file name”);

ifstream class is used to read a stream of object from a file & ofstream class is used to write a stream of objects in a file.

e.g. **ifstream fobj;**
fobj. open (“data.txt”);

File modes :

With class fstream we need to specify file mode.

General form of function open ( ) with a arguments is: --

stream-object.open (“file_name”, mode);

The file mode can take any parameter of the list: -

Parameter	Meaning
1. ios :: app	Append to the end of file
2. ios :: ate	Go to the end of file an opening.
3, ios :: binary	Binary file
4. ios : : in	Open file for reading only.
5. ios :: nocreate	Open fails if the file doesn’t exist.
6. ios :: noreplace	Open fails if the file already exists.
7. ios :: out	Open file for writing only
8. ios :: trunc	Delete contents at the file if it exists.

Mode can combine two or more operators using bitwise OR operator.

e.g, fout.open (“data”, ios :: appl ios :: nocreate);

This opens a file named data in the append mode but fails to open the file if it doesn’t exists.

For ifstream, default mode is ios::in & for ofstream, the default mode is ios::out. For fstream, there is no default mode.

- **closing a file :-**

function close ( ) is used to close a file, which is opened for read, write operations.

It is called automatically, by destructor function. It can be called explicitly. E.g.infile.close();

Binary Files:

In C++ by default the file stream operations are performed in text mode but we can store data in binary form. The functions write() and read() are used for this purpose. The forms of these functions are as follows:

```
in.read(char*)&V,sizeof (V);
out.write(char*)&V,sizeof (V);
```

These functions take two arguments:

i) Address of variable V

ii) Length of that variable in bytes

Example 21:

```
#include<fstream.h>
class A
```

```
{
protected :     char name[40];
int age;
public :
        void getdata ()
        {
cout <<"Enter Name :";
cin >> name ;
cout <<"enter age:";
cin >> age;
}
        void display()
        {
                cout<<"name is:"<<name;
                cout<<"\nage is:"<<age;
        }
};
void main ()
{
A obj;
obj. getdata();
ofstream  fl ("ABC.dat");
fl. write ((char *) & obj,
sizeof (obj)) ;
obj.display();

}
```

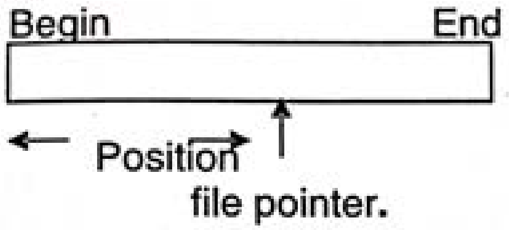
Output:

```
Enter Name :smita
enter age:23
name is:smita
age is:23
Press any key to continue
```

write function takes two arguments:

1. Address of the object to be written.
2. Length of the objects in bytes.
3. We have used size of operator to find the length of object o class.
4. Here the address of the object
5. Must be cast to type pointer to char.

File Pointers:



Each file object has associated with it, 2 integer values called get pointer & put pointer. These are also called current get position & current put position pointers. These values specify the byte number in the file where writing and reading takes place:

There is some situation, when user must take control of the file pointers so that you can read from & write to an arbitrary location in the file.
e.g.seekg( ) and tellg( ) function, allow you to set & examining the get pointers, and the seekp( ) and tellp( ) function perform these same actions on the put pointer.

Specifying the position

Seekg ( ) function set the pointer to the beginning of the file so that reading would start there.

This form of seekg() takes one argument which represents the absolute position in the file.The start of the file is zero.

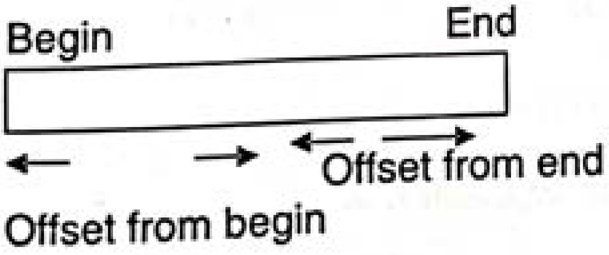
Specifying the offset

seekg( ) function can be used in two ways.

1) One, where the single argument represents the position.
2) You can also use it with two arguments where the first argument represents an offset from a particular location in the file & the 2nd specifies the location on which offset is measured. There are 3 possibilities for the 2nd argument:
    - beg – beginning of the file
    - cur- is the cement, pointer position
    - end - is the end of the file

eg. seekg (-10, ios:: end)
It will set the put pointer to 10 bytes before the end of the file.

seekg( ) fun with two arguments :



Example 22: Program for file manipulation

```
 # include <fstream.h>
# include <stdlib.h>
void main()
{
        const int SIZE = 80;
        char line [SIZE];

        ofstream fout;
        fout.open("country");
        fout << "United States of America\n";
        fout << "United Kingdom        \n";
        fout << "South Korea           \n";
        fout << "India                 \n";
        fout << "Pakistan              \n";
        fout.close();

        fout.open("capital");
        fout << "Washington D.C.\n";
        fout << "London\n";
        fout << "Seoul\n";
        fout << "New Delhi\n";
        fout << "Islamabad\n";
        fout.close();

        ifstream fin1, fin2;
        fin1.open("country");
        fin2.open("capital");
        cout << "Country\t\t\t\tCapital\n\n";
        for (int i = 1; i <= 10; i++)
        {
                if(fin1.eof() != 0)
                {
                        cout << "Exit from country \n";
                        exit(1);
                }
                fin1.getline(line, SIZE);
                cout << line << "\t";
                if(fin2.eof() != 0)
                {
                        cout << "Exit from Capital \n";
                        exit(1);
                }
                fin2.getline(line, SIZE);
                cout << line << "\n";
        }
}
```

Output is :

```
Country                        Capital
United States of America       Washington D.C.
United Kingdom                  London
South Korea                     Seoul
India                          New Delhi
Pakistan                       Islamabad
Exit from country
```

Command –line arguments:

Used when invoking a program from Dos.

They are typically used to pass the name of a data file to an application.

E.g.

```
void main (int argc, char* arg[])
   {
        cout <<”\nNumber of arguments are:”<< argc;
for(int i=0;i<argc; i++)
cout<<”\nArgument “<< i<<”=”<< argv [i];
   }
```

Output when run from the command line is: ---

```
c:\>Prog1 HI Hello Welcome
Number of arguments are: 4 .
argument 0 = c:/cpp/ch14/…/Prog1.exe       (specifies complete path)
argument  1 =HI
argument 2 = Hello
argument 3 =Welcome
```

To read the command line arguments, the main ( ) function itself must be given two arguments.

First is argc. (for argument count) which represents total number of command-line arguments.

The first command-line argument is always the pathname of the current program.

Remaining command-line arguments, are those typed by the user, are delimited by space character

System stores the command-line arguments as strings in memory & creates an array of pointers to this string.

Individual string are accessed through appropriate pointer so the first string (pathname) is argvl0], second is argv [1] and so on……..

SOLVED PROGRAMS

(1) Write the power ( ) function in C++ that returns x raised to the power n, where n can be any integer, double power (double x, int p);
Use algorithm that would compute x^{20} by multiplying 1 by x 20 times. **(March 2002)**

Solution:

```
#include<iostream.h>
   double pow (double, int) ;
void main ( )
       {
              double x;
              int n;
              cin >> x >> n ;
              cout <<x<< "raised to"<<n <<"="<< pow (x, n) << endl;
       }
       double pow (double x, int n)
       {
              if (x==0)
return 0;
              if (n==0)
return 1;
              double y  =  1;
              for (int i  = 0 ; i < n ; i++)
                          y *=  x  ;
              for (i  = 0 ; i > n ; i--)
                          y /=  x ;
              return  y;
       }
```

Output is:

```
2
3
2 raised to 3 =8
```

(2) Write a C++ program to find the greatest common divisor of two numbers. Define a method **find** to accept the values and calculate GCD of two numbers and print the GCD value. **(March 2002)**

Solution:

```
# include<iostream.h>
    class gcd
    {
        int a, b;
        public  :
              void find ( )  ;
    } ;
```

```
void gcd :: find (void)
{
cin >> a >> b ;
while (a != b)
{
        if (a > b)
                a -= b;
                if (b > a)
                b -= a;
}
                cout << a;
}
void main ( )
{
gcd obj;
obj . find ( );
}
```

Output is:

```
15
25
5
```

(3) Write a C++ program that will read a line of text and count the number of words in a text. **(March 2002)**

Solution:

```
# include <iostream · h>
# include <string· h>
void main ( )
{
        int i, len, w = 0;
        char line [80];
        cin . getline (line, 80);
        len = strlen (line);
        for (i = 0; i < len; i ++)
        {
                if (line [i] = = ’ ’)
                    w ++;
        }
        cout << “words = “< w + 1;
}
```

Output:

```
Today is Monday
    :3
```

QUESTIONS

1. Select the correct alternatives

1) Following is not the feature of OOPs

a). Polymorphism b) Inheritance c). Data abstraction d) None of the above.

2) Following operator can't be overloaded:

a) ++ b) :: c) - d) *

3) A mechanism, which automatically destroys" an object when it gets invalid is

a) constructor b) destructor. c). object d) member function

4) Which of the following allows to acceess the private data of other function?

a) Inline function b) Friend function c) main function d) All of the above.

5) Object is defined as:

a) Instance of a class b) "black box" which receives and sends messages.

c) Both of the above. d). None of the above.

6) C++ provides:

a). single inheritance b) multiple inheritance

c) multilevel inheritance d) All of the above

7) A structure is a class all of whose members are by default

a) public b).private c)protected d) none of the above.

8) Which of the following is not a feature of Object Oriented Programming

a)Follows bottom-up approach **(Mar.2004)**

b) Object may communicate with each other through functions

c) Follows top-down approach in design d) Programs are divided into what are known as objects.

9) The ability to take more than one form is called __________ in object oriented programming. **(Oct.2003)**

a) inheritance, b) encapsulation, c) polymorphism, d) data abstraction.

10) __________ is not a derived data type in C++. **(Mar.2005)**

a) Class b) Array c) Function d) Pointer

11) A derived class with several base classes is ______________ inheritance.

a) single b) multiple c) multilevel d) hierarchical **(Mar.2006,09)**

12) Which of the following allows to access the private data of other class is

a) In-line function b) Friend function **(Mar.2007)**

c) Main function d) All of the above

13) Following operator cannot be overloaded ______________. **(Mar.2008)**

a) + + b) : : c) - d) *

14) For a=23 and b=3 the value of c after execution of c=(a/b)*(a%/b)is
a) 14 b) 49 c) 21 d)69 **(Mar.2010)**

15) In C++ double type data consumes ——bytes in memory **(Mar. 2016)**
a) 2 b) 4 c) 6 d)8

16) If all visibility labels are missing then by default members of class are ——
a) public b) private c)protected d) Void **(Mar. 2017)**

17) What will be the value of x after execution of following expression in C^{++}?
X = ++ m + n + +; where m = 10 and n = 15. **(Mar. 2019)**
a) 25, b) 27, c) 26, d) 28

18) ______ operator cannot be overloaded. **(Mar. 2020)**
a) ++ b) + c) :: d) >>

THEORY QUESTIONS

1. What is C++ ? write advantages of C++. **(Mar.2003)**
2. Explain the difference between procedural programming and object oriented programming. **(Specimen Paper)**
3. Write any six main advantages of OOPs. **(Mar.2003)**
4. Explain classes and objects.
5. What is resemblance between arrays and pointers?
6. Why references are used?
7. What is difference between class and structure? **(Specimen Paper)**
8. What is runtime and compile time polymorphism? How it is achieved? **(Specimen Paper)**
9. What is operator function? Describe the syntax of operator function. **(Mar. 2017)**
10. Explain the difference between operator function as a member function and as a Friend function. **(March 2002,2016)**
11. Explain difference in function overloading and function overriding.
12. Explain virtual function. Why it is necessary?
13. State any 8 characteristics of virtual functions. **(March 2002)**
14. Explain difference between nested classes and derived classes.
15. What is class? Give general form of class declaration. **(Mar.2003)**
16. State characteristics of friend function **(March 2016,2017)**
17. Explain constructor and destructor with example. **(March 2002)**
18. What is a Constructor in C++? With a suitable example explain how a constructor is declared and defined. **(March 2005)**
19. Explain how memory address of a variable is accessed in C++. **(Model Paper, Mar.04)**

20. Why there is necessity of overloading an operator. **(Model Paper)**

21. Explain with a diagram, the file input and file output streams in C++.**(Mar.2006)**

22. Enlist the basic data types used in C++ with size of data in terms of bytes for each. **(March 2006)**

23. Explain how the memory address of a variable can be accessed in C++. **(March 2007)**

24. What are pointers in C++? Explain the use of pointer variables for function definition using call by value and call by reference. **(March 2007,08,09,19)**

25. What is the function of each of the following file stream classes?

    i) ifstream ii) ofstream iii) filebuf ++. **(March 2008,2020)**

26. What is inheritance in C++? What are different forms of inheritance? Give example for each. **(Mar.2003)**

27. What is the difference between function overloading and function overriding?

28. What is runtime and compile time polymorphism? How it is achieved? **(Specimen Paper)**

29. What is operator overloading? State the three steps involved in the process of overloading. **(Mar.2003,20)**

30. What is a Constructor and a Destructor? Give one example for each. **(Mar.2006,09,2020)**

31. What are Destructors?Write the rules for writing destructor function. **(Mar.2009)**

32. Explain Runtime and Compile time polymorphism? Give any one example. **(Mar.2009,2017)**

33. State any six characteristics of constructor function. **(Mar.2003,2016,2019)**

34. Explain the use of scope resolution operator and memory management operators in C++ with examples. (Mar.2004)

35. What are classes in C++ for file stream operation? How do you open and close files in C++? Explain any four file modes. **(Mar.2004)**

36. Explain the following concepts with an example of each:

    a) Inheritance b)Polymorphism c) Data Abstraction **(Mar.2004)**

37. What are classes in C++ ? How are member functions defined inside and outside the class? Explain with examples. **(Mar.2004)**

38. Explain basic and user defined types in C++. **(March 2002)**

39. Explain the tree types of data conversion in C++ with a suitable example. **(March 2005,2019)**

40. Explain local data and global data variables in C++ using example. **(Mar.2016)**

41. Using Examples explain how files are opened and closed in C++ ? state any four file modes **(Mar.2016)**

4233.State three characteristics of static data. **(Mar.2016)**

43. Explain use of memory management operators in C++ . **(Mar.2016)**

44.Explain the concept of function overloading with example. **(Mar. 2017)**

45. Explain the use of scope resolution operator and memory management operators in C++ with examples. **(Mar. 2017)**

46. Explain the syntax of C++ program structure with example. **(Mar. 2019)**

47. Explain any six operators used in C++. **(Mar. 2019)**

48. Explain friend function in C++ with example. **(Mar. 2020)**

49. Write C++ declaration for the following. **(Mar. 2020)**

i) Array of 10 integers ii) Pointer to character variable iii) Object of the class

PROGRAMMING EXERCISE

1. Write a C++ program that right justifies text. It should read and echo a sequence of left justified lines and print then in right justified format. **(Mar.2006)**

2. Implement a class temperature to convert degree Fahrenheit Value to degree Celsius Value. [Hint : C/5 = F – 32/9, where C is temperature in degree Celsius and F is temperature in Fahrenheit degree] **(Mar.2006,09,16)**

3. Write a program in C++ to find the Greatest common Divisor (GCD) of two natural numbers. **(Mar.2006)**

4. Write a C++ program to display first 20 terms of Fibonacci series. **(Mar. 2007,09,16)**

5. Write a C++ program to find factorial of a natural number inputted during program execution. **(Mar. 2008)**

6. Write a program in C++ that right justifies text. It should read and echo a sequence of left justified lines and print them in right justified format. **(Mar.2007)**

7. Write a C++ program to replace every space in an inputed string (less than 80 characters) with a hyphen (i.e. -) **(Mar. 2007)**

8. Implement a Circle Class. Each object of this class will represent a circle accepting its radius value a float. Include an area ( ) function which will calculate the area of circle. **(Mar. 2008)**

9. Write a C++ program to find greatest common divisor of two numbers. Define a method find to accept the values and calculate GCD of two numbers and print the GCD value. **(Mar. 2008,2011,17)**
10. Write a C++ program that will read a line of text and count the number of words in a text. **(Mar. 2008)**
11. Write a C++ program to count occurrence of a character 'J' in a string.
12. Write a C++ program by using swap function to interchange two given numbers. **(Mar. 2010)**
13. Write a C++ program to accept a number and test whether it is a Prime number. **(Mar. 2011)**
14. Write an object Oriented program in C++ to read an integer number and find out the sum of its all digits.(e.g. if n=1256 SUM = 1+2+5+6) **(Mar. 2011)**
15. Implement a class average . Include a constructor in it which will accept value of three variables from user. Include two more functions in it, one calculates average and other prints it. **(Mar. 2016)**
16. Write a C++ program to find factorial of entered number. **(Mar. 2017)**
17. Write a C++ program to accept a sentence (maximum 50 characters) and print sentence in reverse. **(Mar. 2019)**
18. Write a function in C++ to accept four integers. Find the smallest integer and print it. **(Mar. 2019)**
19. Write a C++ program to find smallest in an array of 10 floats using pointer. **(Mar. 2019)**
20. Write a class based program in C++ to find area of a Triangle. **(Mar. 2019)**
21. Write a C++ program to accept 10 integers in an array and find its sum and average. **(Mar. 2020)**
22. Write a C++ program to accept a sentence of 80 characters and count number of word in a sentence. **(Mar. 2020)**

Answers Q.(1)

1) None of the above, 2) ::, 3) Destructor, 4) Friend function
5) Both of the above, 6) All of the above, 7) Public
8) Follows bottom-up approach 9) polymorphism 10) Class
11) multiple 12) Friend function 13) : : 14)14 15) 8
16) private 17)26 18) ::

❑❑❑

4
HTML

INTRODUCTION

HTML is an evolving language, and stands for **Hyper Text Markup Language**

- It is a text file containing small **markup tags** which are instructions given to browser about **how to display** the page.
- An HTML file must have an **htm** or **html** file extension.
- An HTML file can be created using a **simple text editor** (like Notepad in windows).

HTML was invented by Tim Berners-Lee while at CERN, the European Laboratory for Particle Physics in Geneva.

4.1 WHY HTML

Hypertext is an ordinary text with formatting facilities and Markup is the process of taking ordinary text and adding extra features to it.

Thus the HTML documents are text files made up of HTML elements that define a document and guide its display. In practical terms, HTML is a collection of *platform-independent styles* that define the various components of a World Wide Web document. HTML provides the user with a consistent interface and a highly effective medium for presenting information to developers.

HTML is a set of instructions given to Web browser for formatting and layout of WebPages, so it may be possible that a look of the same HTML code may differ since different browsers may interprets them differently.

Advantages of HTML

1) HTML is an easy to use, learn, implement and flexible alternative to traditional presentation and tedious software.
2) Contains powerful formatting facilities.
3) HTML documents are device and platform independent. (Since it can be designed to work on not only home PCs but also on graphical workstations, dumb terminals, network computers, hand-held devices etc.)
4) You can traverse to any HTML document required because of hyper linking facility available, thus controlled navigation is possible.
5) Required HTML pages can be updated easily, without changing whole document.
6) It is a kind of software, which has been called world ware.
7) Independent work can be done and you need not rely on application or program vendor.
8) No expensive license software or hardware required.

9) If compatibility with user habits, expectations and multiple platforms is the goal, then HTML is the only approach to delivering a web application.

Disadvantages of HTML

1) HTML doesn't offer programming languages features and capabilities.
2) It's easy to write "bad" HTML containing errors.
3) Complex HTML code is hard to read and understand and code complexity increases to make interactive web page. So building complex pages is very time consuming.
4) It's easy to make mistakes (e.g. leaving out a ">" or "/" character).
5) Special types of software like scripting languages (VB Script, Java Script) are required for handling different events and validations.
6) Can't detect errors easily since no special debugging tool is provided.

4.2 HTML DOCUMENTATION

HTML documents are plain-text (also known as ASCII) files that can be created using any text editor (e.g., Emacs or vi on UNIX machines; Simple Text on a Macintosh; Notepad on a Windows machine). You can also use word-processing software if you remember to save your document as "text only with line breaks".

HTML Editors

HTML Editors are programming tools for Hyper Text Markup Language (HTML) documents.

There are three categories of HTML Editors:

1. Text Editors
2. HTML Code Editors
3. HTML Design Tools

1) Text Editors

These editors only edit ASCII text. They offer no functionality to facilitate better HTML development. They are useful if your knowledge of HTML is excellent. Some examples of Text Editors include Notepad (Windows), Simple Text (Macintosh), and Pico (Unix).

They are typically WYSIWYG. WYSIWYG is an acronym for "what you see is what you get"; it means that you design your HTML document visually, as if you were using a word processor, instead of writing the markup tags in a plain-text file and imagining what the resulting page will look like.

2) HTML Code Editors

These editors may or may not be WYSIWYG.

3) HTML Design Tools

These tools are intended for HTML development without exposing the code to the author. They are typically WYSIWYG. Many of these tools do allow the user to access the HTML code, however this is not usually apparent to the user. Some examples of HTML Design Tools include NetObjects Fusion (Web Development), and Microsoft Office 97 (Traditional Office/Design Tools that provide HTML output).

You can concentrate on the content, rather than the syntax, of your Web site, You can Create a Web site without learning HTML, You can Design Elegant and Consistent Web sites with a few key strokes, since they are more user friendly

➢ HTML ELEMENT

An ***element*** is a fundamental component of the structure of a text document. Some examples of elements are heads, tables, paragraphs, and lists. Elements can contain plain text, other elements, or both.

HTML elements are defined using HTML tags.

➢ HTML Tags

- HTML tags are used to mark-up HTML **elements.**
- HTML tags are surrounded by the **two characters < and >**
- The surrounding characters are called **angle brackets.**
- HTML tags normally **come in pairs** like and they are usually paired to start and end the tag instruction.
- The first tag in a pair is the **start tag,** the second tag is the **end tag.**
- The text between the start and end tags is the **element content.**
- HTML tags are **not case sensitive;** <b> means the same as <B>.

  Syntax is:

  <tag name> text (element content) </tag name>

  <tag name attribute name="argument/value"> text (element content) </tag name>

  or just

  <tag name>

For example:

<TITLE> WELCOME TO MY WEBSITE </TITLE>

<A HREF="URL"> CLICK HERE </A>

An HTML document is composed of a single element:

<HTML> . . . </HTML>

that is, in turn, composed of head and body elements:

<HEAD> . . . </HEAD>

and <BODY> . . . </BODY>

➢ Tag Attributes

Tags can have attributes. Attributes can provide additional information about th HTML elements on your page, included inside the start tag.

Attributes always come in **name/value pairs** like this: name="value"

Syntax is

<tag name attribute name="value">

e.g. <IMG SRC="c:\a.jpg">

so here src is an attribute for image tag. Normally its value is specified in quotes.

• Some useful Tips while writing HTML document

HTML is **not case sensitive.** *So* <head> is equivalent to <HEAD> or <Head>

Also **all tags are not** supported by all World Wide Web browsers. If a browser doe not support a tag, it will simply ignore it.

When you write HTML text, you can never be sure how the text is displayed i another browser. The text will **be reformatted every time** the user resizes hi window. Never try to format the text in your editor by adding empty lines and space to the text.

HTML will **truncate the spaces** in your text. Any number of spaces counts a one. In HTML a new line counts as one space.

Using empty paragraphs <p> to insert blank lines is a bad habit. Use the <br> tag instead. You might have noticed that paragraphs can be written without the closing tag </p>. Don't rely on it. The next version of HTML might not allow you to skip any closing tags.

HTML automatically **adds an extra blank line** before and after some elements, like before and after a paragraph, and before and after a heading.

4.3 THE MINIMAL HTML DOCUMENT

Every HTML document should contain certain **standard HTML tags.** Each document consists of head and body text. The head contains the title, and the body contains the actual text that is made up of paragraphs, lists, and other elements. Browsers expect **specific information** because they are programmed according to HTML and SGML specifications.

➢ Simple html document

```
<HTML>
        < HEAD >
                <TITLE> HOME PAGE
                </TITLE>
        </HEAD>
        <BODY> THIS IS MY FIRST WEB PAGE
        </BODY>
</HTML>
```

(When you save an HTML file, you can use either the .htm or the .html extension. Save the file as "firstpage.htm". Open this file through your Internet browser.)

Example Explained

The first tag in your HTML document is <HTML>. This tag tells your browser that this is the start of an HTML document. The last tag in your document is </HTML>. This tag tells your browser that this is the end of the HTML document. The text between the <HEAD> tag and the </HEAD> tag is header information. Header information is not displayed in the browser window. The title is displayed in your browser's caption.

The text between the <BODY> tags is the text that will be displayed in your browser.

The text between the <TITLE> tags is the title of your page

4.4 BASIC HTML TAGS

The most important tags in HTML are tags that define headings, paragraphs and line breaks.

1.<HTML>

The HTML tag identifies a document as an HTML document. All HTML documents should start with the <HTML> tag and end with the </HTML> tag.

Syntax

<HTML>…….</HTML>

e.g.

The following example begins and ends a short document with the HTML tag.

```
<HTML>
 <BODY>
  This is HTML file.
 </BODY>
</HTML>
```

Following tags appear in <HTML>tag.:--

- **HEAD**

The HEAD contains **general information**, or *meta*-information, about the document. The HEAD tag defines an HTML document **header**. It is the first thing in any document, lying above the BODY and just after the <HTML> tag starting the document. The contents of the HEAD are not displayed as part of the document.

The HEAD tag can contain TITLE, BASE, ISINDEX, META, SCRIPT, STYLE, and LINK tags.

Syntax

<HEAD>...</HEAD>

e.g.
```
<HTML>
        <HEAD>
                <TITLE>WELCOME TO FIRST WEBSITE</TITLE>
        </HEAD>
       </HTML>
```

- **TITLE**

  The title of a document is specified by the TITLE element, which should be placed in the document HEAD. Each document can have **only one title**, which should identify the document content in a general way.

  The Title **is *not* part of the document text** and cannot contain hypertext links or special markup commands -- it must be simple text. Often the title is used to label the window displaying the text, or is used to label a place in a browser's history or bookmark list. It therefore should be short -- less than 64 characters

Other tags can be placed are:

- **STYLE** -- Stylesheet instructions, written in a stylesheet language. Stylesheet instructions specify how the document should be formatted for display. Very few browsers currently support stylesheets.
- **SCRIPT** -- Script program code -- for enclosing, within a document, scripting program code that should be run with -- and that can interact with -- the document. Example languages are JavaScript and VBScript.

E.g.
```
<HTML>
  <HEAD>
    <SCRIPT language="VBSCRIPT"> VBscript </SCRIPT>
  </HEAD>
  <BODY>
     ..... text of the document
  </BODY>
</HTML>
```

2.<BODY> Defines the document's body

The BODY element contains **all the contents** of a document

Various mark-up elements are allowed within the body to indicate headings, paragraphs, lists, hypertext links, images, and so on.

➢ **Attributes in body tag**

1) The **BACKGROUND** Attribute

This allows you to specify an image file to use as a background (a bit like a watermark) behind the displayed text and graphics.

E.g.
```
<BODY BACKGROUND="c:\a.gif">
        Text….
   </BODY>
```

So image a.gif will be set as a background to your web page.

2) **Background color** of the web page
Attribute is:BGCOLOR="#rrggbb"

Sets the background color to the specified RGB color value, where RR GG and BB are the hexadecimal color codes for the Red, Green and Blue levels, ranging from 0 to 255 -- that is, 00 to FF. The color "000000" is black, while "FFFFFF" is white.

3) Setting the **text color(TEXT** Attribute)

Syntax: <BODY TEXT="#rrggbb">text in a body</BODY>
Sets the default text color to the specified RGB color value.

4) Setting color for hyperlinks (**LINK** Attribute)
Syntax: <BODY LINK="#rrggbb">text in a body</BODY>
Sets the default text color of hypertext anchors to the specified RGB color value.

5) Setting color for visited hyperlinks (**VLINK** Attribute)

Syntax: <BODY VLINK="#rrggbb">text in a body</BODY>

Sets the default text color of *visited* hypertext links to the specified RGB color value.

➢ **Elements in the BODY are categorized as**

A) Text Block Elements

BODY element contains all the displayed content of a document. Structurally, the document content is organized into blocks of text, such as paragraphs, lists, headings, paragraphs, block quotations, and so on. These are generically called *block elements*, since they "block" chunks of text together into logical units. Block elements can often contain other blocks -- for example, a list item can contain paragraphs or block quotations, so that these elements can often nest together.

The block-level elements are:

- Hn (Headings) (h1 to h6)
- P
- ADDRESS
- BLOCKQUOTE
- PRE
- HR
- FORM
- TABLE

B) **Text Emphasis Elements**

These are elements that mark text for special meanings, for example, that a particular piece of text is emphasized (EM) or a citation (CITE), or that specifies the desired physical formatting, such as boldface (B) or italics (I). These elements can usually appear anywhere inside a block element, with a few exceptions (you can't have images inside a PRE element).

C) **Special Elements** -- Hypertext Anchors

Analogous to the text-level markup is the anchor (A) element. This is the element that marks hypertext links. Obviously you want to know a lot about this one.

D) **Character-Level Elements**

Then are what I call character-level elements, namely line breaks (BR) and images (IMG). These are treated much like characters, and can appear wherever there is a character in a document.

E) **Character References**

Finally there are character or entity references. These are special HTML "escape" codes that can be used to enter special characters that are hard to type, such as accented or other non-ASCII characters. You also need to use these to type angle brackets or ampersand characters -- as these are otherwise interpreted as HTML tags (< ... >) or as the beginnings of character or entity references (&).
Analogous to the text-level markup is the anchor (A) element. This is the element that marks hypertext links. Obviously you want to know a lot about this one.

3. <P> Defines a paragraph

Paragraphs are defined with the <P> tag. Can contain align attribute for alignment of the text within paragraph.
E.g.<P ALIGN="CENTER"> this is a paragraph</P>

HTML automatically adds an extra blank line before and after a paragraph. You must indicate paragraphs with <P> elements. A browser ignores any indentations or blank lines in the source text. Without <P> elements, the document becomes one large paragraph. The </P> closing tag may be omitted. This is because browsers understand that when they encounter a <P> tag, it means that the previous paragraph has ended. However, since HTML now allows certain attributes to be assigned to the <P> tag, it's generally a good idea to include it.

- **ADDRESS Element**

The ADDRESS element is used for address information, signatures, statements of authorship, etc. It is often placed at the bottom (or top) of a document. The rendering of the contents of the ADDRESS is left up to the browser -- most browsers render the ADDRESS in italics. It may also be right justified, or indented.

e.g.

```
<ADDRESS><A HREF="c:\add.html">X.Y.Z.</A></ADDRESS> </P>
<P><ADDRESS>
WEB DESIGNER<BR>
Tel (023) 122 123.
</ADDRESS> </P>
```

These are rendered as

X.Y.Z
WEB DESIGNER
Tel (023) 122 123.

An address cannot contain P, BLOCKQUOTE, FORM or other block elements, but can contain text, text markup (emphasis, etc.), anchor elements or even images.

4. <BR> Inserts a **single line break**

Line Breaks:

The <BR> tag is used when you want to end a line, but don't want to start a new paragraph. The <BR> tag forces a line break wherever you place it.

E.g. This <BR> is line breaks.

So here This and is line breaks will appear on two different lines.

The <BR> tag is an empty tag. It has no closing tag.

5. <HR> Defines a **horizontal rule.**

Used to produce a horizontal line, the width of browser's window.
It allows to differentiate sections of your document.
SIZE and WIDTH attributes will let you alter the thickness
And the percentage of the windows covered by it.

e.g.

```
<BODY>
        <H1>This is my first web page</H1>
        <HR SIZE=5 WIDTH="20%">
   </BODY>
```

6. <! Defines a comment in the HTML source code Comments in HTML.

The browser will ignore a comment. You can use comments to explain your code, which can help you when you edit the source code at a later date.

E.g.<!-- This is a comment -->

Note that you need an exclamation point after the opening bracket, but not before the closing bracket.

Comments **do not nest**, and the double-dash sequence "--" may not appear inside a comment except as part of the closing --> tag. You must also make sure that there are no spaces in the start-of-comment string.

For example, the line

```
<!--  This is a valid comment -->
< !--  This is not a valid comment -->
```

is not, since there is a space between the left angle bracket and the exclamation mark.

7. Headings

<H1>---<H6> Defines heading 1 to heading 6.

Headings are defined with the <H1> to <H6> tags. <H1> defines the largest heading. <H6> defines the smallest heading (as shown).

Heading Size 1	Heading Size 2
Heading Size 3	Heading Size 4
Heading Size 5	Heading Size 6

E.g.

<H1 STYLE="COLOR: blue">THIS is displayed in large font with blue color </H1>
<H1 STYLE="FONT-FAMILY:verdana">This is displayed in verdana font style </H1>
<H1 STYLE="FONT-SIZE:150%">You can define size in percentage also.</H1>

HTML automatically adds an extra blank line before and after a heading.

8. <PRE> tag

This tag preformats the text. The text appearing between <pre> and </pre> is displayed in monospace form. Using this tag, we can position the characters. It can also be used for columnar lists.

Eg. <PRE> Employee_name Employee_number Employee_address </PRE>

So output is displayed as :

```
Employee_name          Employee_number          Employee_address
```

4.g. FONT TAGS IN HTML

To have different size and color to the text, rather than using headings tags, many people uses **<FONT>tag.**

The <FONT> tag in HTML is deprecated. It is supposed to be removed in a future version of HTML.

Font attributes:

Attribute	Example	Purpose
size="number"	size="2"	Defines the font size
size="+number"	size="+1"	Increases the font size
size="-number"	size="-1"	Decreases the font size
face="face-name"	face="Times new Roman"	Defines the font-name
color="color-value"	color="#eeff00	Defines the font color
color="color-name"	color="red"	Defines the font color

e.g.

```
<P>
<FONT SIZE="5" FACE="VERDANA" COLOR="#AA00FF">
This is example of demonstrating fonts. </FONT>
</P>
```

- **<MARQUEE> TAG**

MARQUEE, supported only by the Microsoft Internet Explorer 2 (and later) browser, is used to create a scrolling text marquee.

e.g. <MARQUEE ALIGN="top">Scrolling text </MARQUEE>
creates a text marquee with the enclosed text scrolling along the frame.

The another attribute is **DIRECTION** defining direction of the marquee text.
e.g.<MARQUEE DIRECTION="RIGHT">WELCOME</MARQUEE>

So here WELCOME scrolls from left towards right. The default direction is right to left.

But other commercial browsers, including Netscape Navigator, do not support this element.

➢ Character Formatting

HTML has two types of styles for individual words or sentences: logical and physical. **Logical styles** tag text according to its meaning, while *physical styles* indicate the specific appearance of a section. For example, in the preceding sentence, the words "logical styles" was tagged as "strong." The same effect (formatting those words in italics) could have been achieved via a different tag that tells your browser to "put these words in bolds." Try to be consistent about which type of style you use.

Logical Styles:

<EM>

for emphasis. Typically displayed in italics. (*Consultants cannot reset your password unless you call the help line.*)

<CODE>

for computer code. Displayed in a fixed-width font. (The <stdio.h> header file)

<STRONG>

for strong emphasis. Typically displayed in bold. (**NOTE:** Always check your links.)

<VAR>

for a variable, where you will replace the variable with specific information. Typically displayed in italics. (rm *filename* deletes the file.)

Physical Styles

<B> bold text :It boldface text appearing between <B> and </B> tag.
e.g.<B>In Bold</B>
Output is **In Bold**

<I>italic text : It italicize text appearing between <I> and </I> tag.
e.g.<I>In Italics</I>
Output is ***In Italics***

<U>underline text : It underlines text appearing between <U> and </U> tag.
e.g.<U>Underlined</U>
Output is **Underlined**

- **<SUB> tag:** It is subscript tag. The text enclosed in <SUB>and </SUB> is displayed in subscript form. This is useful in representation of mathematical formulae or chemical equations.
  e.g. H<SUB>2</sub>SO<SUB>4</SUB>
  output displayed as :H_2SO_4

- **<SUP> tag:-** It is superscript tag. The text enclosed in <SUP>and </SUP> is displayed in superscript form. This is also useful in representation of mathematical formulae or chemical equations.
  e.g. volume of sphere is=(4/3)(3.14)r<SUP>3</SUP>
  Output displayed as: volume of sphere is=$(4/3)(3.14)r^3$

4.6 DEALING WITH URLS

Hypertext Anchors

URL stands for **Uniform Resource Locator,** which may represent an address of document on web or Internet or simply a path to a document in a specific directory. The World Wide Web uses Uniform Resource Locators (URLs) to specify the location of files on other servers. A URL includes the type of resource being accessed (e.g., Web, gopher, FTP), the address of the server, and the location of the file. The syntax is:

scheme://host. domain [*:port*]*/path/ filename*

HTML provides you to jump from a link to any document or image or any local or WebPages by using special tag, called <a> i.e. Anchor tag.

Syntax is: <A HREF="URL">name or image which can be treated as link</A>

Thus ***anchor*** is a piece of text or some other object (e.g. image), which marks the beginning and/or the end of a hypertext link. The <A> element is used to mark that piece of text (or inline image), and to give its hyper textual relationship to other documents. The text between the opening and closing tags, <A attributes> ...text... </A> can act as start or destination (or both) of a link. The HREF attribute (which is actually optional) marks the anchor as the start of a link *to* another document or resource (it could point, for example, to an image file), or to a particular place in another document.

Syntax is:

```
<A HREF="URL (absolute or relative path)">anchor name </A>
```

An absolute or partial URL can specify the address of the referenced document:

e.g.

1) Link to a page on the World Wide Web.

```
<A HREF="http://www.yahoo.com">Enter your email-id</A>
```

The string ` Enter your email-id ' is a hypertext link *to* the website indicated by URL specified

2) Link to an image by image as a link.

```
<A HREF="image2.jpeg"> <IMG SRC="image1.gif"> </A>
```

The image `image1.gif' is a hypertext link to the image file located in the same directory This will allow you to use a small icon that links the user to a larger version of the same image.

3) Link to document located in different directory

```
<A HREF="d:\soft\a.html"> Click Here.</A>
```

Here by clicking on Click Here link, destination page will be displayed which is specified in the path given.

4) Link to the same page(Links to a Particular Place in a Document)

```
<P><A HREF="#samepage">This is link to the same page. </A></P>
<A NAME="samepage"><H2>Yes You are in the same page.</H2></A>
```

Particular places in an HTML document can be marked as specific destinations of hypertext links via the NAME attribute. Links to Specific Sections

Anchors can also be used to move a reader to a *particular section* in a document (either the same or a different document) rather than to the top, which is the default. This type of an anchor is commonly called a ***named anchor*** because to create the links, you insert HTML names within the document.

5) Links Between Sections of Different Documents

Suppose you want to set a link from document A (fileA.html) to a ***specific section*** in another document (fileB.html).

Enter the HTML coding for a link to a named anchor:

- fileA.html:--

```
<P>
This is link
<A HREF=" fileB.html #differentlink">referenced to a specific section </A> in
different document.
</P>
```

Think of the characters after the hash (#) mark as a tab within the fileB.html file. This tab tells your browser what should be displayed at the top of the window when the link is activated.

- Next, create the *named anchor* (in this example "differentlink ") in fileB.html as:

<H2><A NAME=" differentlink "> to a specific section </A></H2>

With both of these elements in place, you can bring a reader directly to , " to a specific section"word in fileB.html

Important Hint

You cannot make links to specific sections within a different document unless either you have write permission to the coded source of that document or that document already contains in-document named anchors

Attributes for A and LINK :→The following are the attributes appropriate to either Anchor or LINK elements. The other attributes are less common, and can be omitted at an introductory reading.

1. HREF (link to object)
2. NAME (link from object
3. TITLE (TITLE of document)
4. METHODS (how to link -- this attribute has been dropped from the HTML specification, and should not be used)

6) Link to Mail (Mailto)

You can make it easy for a reader to send electronic mail to a specific person or mail alias by including the **mailto attribute** in a hyperlink.

Syntax is:

<A HREF="mailto:*emaiid@host*">*Name*</A> .

e.g.

<A HREF="mailto:spc@*yahoo.co.in*">Mail me</A>

4.7 LISTS IN HTML

HTML supports **ordered, unordered and definition** lists.

Different List Tags are:

<OL>	Defines an ordered list
<UL>	Defines An Unordered List
<LI>	Defines a list item
<DL>	Defines a definition list
<DT>	Defines a definition term
<DD>	Defines a definition description
<DIR>	Deprecated. Use <UL> instead
<MEN>	Deprecated. Use <UL> instead

1.Unordered Lists

An unordered list is a list of items. The list items are **marked with bullets** (typically small black circles). To make an unnumbered, bulleted list,

1. Start with an opening list <UL> (for unnumbered list) tag Enter the <LI> (list item) tag followed by the individual item; no closing </LI> tag is needed
2. End the entire list with a closing list </UL> tag

Inside a list item you can put paragraphs, line breaks, images, links, other lists, etc.

```
e.g. <HTML>
   <BODY>
         <H4>This is Unordered List</H4>
         <UL type = "circle">
            <LI>Sunday</LI>
            <LI>Monday</LI>
            <LI>Tuesday</LI>
         </UL>
   </BODY>
     </HTML>
```

so output given by browser is:

This is Unordered List:

- Sunday
- Monday
- Tuesday

By specifying type attribute in the <UL> tag you can change shape of the bullet. The standard shapes provided are "disk", "circle", and "square"

2.Ordered Lists

An ordered list is also a **list of items**. The list items are **marked with numbers**. A ordered list also called an *numbered* is identical to an unnumbered list, except it uses <OL> instead of <UL>. Inside a list item you can put paragraphs, line breaks, images, links, other lists, etc. You can specify style of numbering for the list items by giving type attribute in <OL> tag and it can take values "I" for uppercase roman, "i" for lower case roman, "A" for uppercase letters, "a" for lower case alpha numeric letter. The start attribute in <OL> tag is used to start the list from the required number. e.g. <OL start = "6"> will start the list items from number 6.

```
e.g. <HTML>
     <BODY>
          <H4><B>This is an Ordered List</B></H4>
          <OL type = "i">
            <LI>Sunday</LI>
            <LI>Monday</LI>
            <LI>Tuesday</LI>
          </OL>
```

```
        </BODY>
     </HTML>
```

so output given by browser is:

This is an Ordered List:

i) Sunday
ii) Monday
iii) Tuesday

3.Definition Lists

A definition list is **not a list of items**. This is a list of terms and explanation of the terms. A definition list starts with the <DL> tag. Thus a definition list (coded as <DL>) usually consists of alternating a *definition term* (coded as <DT>) and a definition description (coded as <DD>). Web browsers generally format the definition on a new line and indent it.

e.g.

```
  <HTML>
     <BODY>
           <H4>This is a Definition List</H4>
                 <DL>
                    <DT>VB</DT>
                         <DD>Programming Language</DD>
                    <DT>VB SCRIPT</DT>
                         <DD>Scripting Language </DD>
                 </DL>
     </BODY>
        </HTML>
```

so output given by browser is:

This is a Definition List:

VB
        Programming Language
VB SCRIPT
        Scripting Language

Inside a definition-list definition (the <DD> tag) you can put paragraphs, line breaks, images, links, other lists, etc.

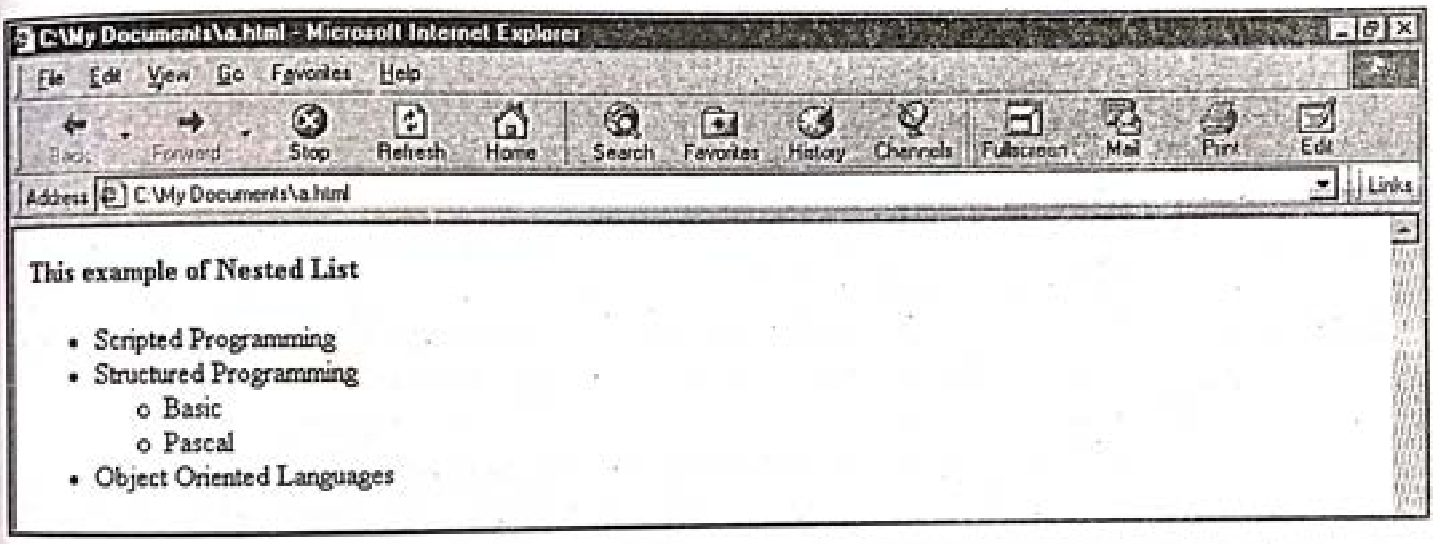
4.Nested lists

Lists can be nested. You can also have a number of paragraphs, each containing a nested list, in a single list item.

e.g.

```
  <HTML>
     <BODY>
      <H4>This example of Nested List</H4>
```

```
    <UL>
        <LI>Scripted Programming</LI>
        <LI>Structured Programming
         <UL>
            <LI>Basic</LI>
            <LI>Pascal</LI>
         </UL>
        </LI>
        <LI>Object Oriented Languages</LI>
    </UL>
  </BODY>
</HTML>
```


C:\My Documents\a.html - Microsoft Internet Explorer
File
Edit
View
Go
Favorites
Help
Back
Forward
Stop
Refresh
Home
Search
Favorites
History
Channels
Fullscreen
Mail
Print
Edit
Address
C:\My Documents\a.html
Links

This example of Nested List

- Scripted Programming
- Structured Programming
  - Basic
  - Pascal
- Object Oriented Languages

4.8 TABLES IN HTML

The TABLE tag defines a table. Inside the TABLE tag, use the TR tag to define rows in the table, use the TH tag to define row or column headings, and use the TD tag to define table cells. The TABLE tag can also contain a CAPTION tag, which specifies the caption for the table you can specify the width of the border surrounding the table and the default background color of the table. (Individual rows and cells in the table can have their own background color.) You can use the CELLSPACING attribute to specify the distance between cells in the table and the CELLPADDING attribute to specify the distance between the border and content of every cell. If you specify the width and height of the table, the browser will do its best to make the table fit the specified dimensions, but in some cases this may not be possible. For example, if the table contains cells that contain non-wrapping long lines, the table may not fit in a specified width.

Syntax :

```
<TABLE
 ALIGN="LEFT|RIGHT|CENTER"
 BGCOLOR="color"
 BORDER="value"
 CELLPADDING="value"
```

```
  CELLSPACING="value"
  HEIGHT="height"
  WIDTH="width"
  COLS="numOfCols"
  HSPACE="horizMargin"
  VSPACE="vertMargin"
>
...
</TABLE>
```

- **Table Elements**
  - <TABLE> ... </TABLE>→ defines a table in HTML. If the BORDER attribute is present, your browser displays the table with a border
  - <CAPTION> ... </CAPTION> →defines the caption for the title of the table. The default position of the title is centered at the top of the table. The attributes ALIGN=BOTTOM can be used to position the caption below the table.
  - <TR> ... </TR>→ specifies a table row within a table. You may define default attributes for the entire row: ALIGN (LEFT, CENTER, RIGHT) and/or VALIGN (TOP, MIDDLE, BOTTOM).
  - <TH> ... </TH> →defines a table header cell. By default the text in this cell is bold and centered. Table header cells may contain other attributes to determine the characteristics of the cell and/or its contents.
  - <TD> ... </TD> →defines a table data cell. By default the text in this cell is aligned left and centered vertically. Table data cells may contain other attributes to determine the characteristics of the cell and/or its contents.

- **Table Attributes**
  - **ALIGN**
    Specifies the horizontal placement of the table.
    LEFT aligns the table on the left (the default).
    RIGHT aligns the table on the right.
    CENTER aligns the table in the center

  - **BGCOLOR="color"**
    sets the color of the background for the table. This color can be overridden by a BGCOLOR tag in the TH, TR, or TD tags.

  - **BORDER="value"** indicates the thickness, in pixels, of the border to draw around the table. Give the value as an integer. no border. means value 0

  - **CELLPADDING="value"**
    determines the amount of space, in pixels, between the border of a cell and the contents of the cell. The default is 1.

- **CELLSPACING="value"** determines the amount of space, in pixels, between individual cells in a table. The default is 2.

- **HEIGHT="height"**
  specifies the height of the table. The default is the optimal height determined by the contents of each cell. Can be a number of pixels, or a percentage of the height of the page or parent element.

- **WIDTH="width"**
  defines the width of the table. The default is the optimal width determined by the contents of each cell. Can be a number of pixels, or a percentage of the height of the page or parent element.

- **COLS="numberOfColumns"**
  Indicates how many virtual columns of equal width fit in the width of the window. If the WIDTH attribute is supplied, the COLS attribute indicates how many virtual columns fit in the specified width. Suppose that the WIDTH attribute is "80%" and the COLS attribute is 4. In this case, each column takes up 20% of the width of the window. Note, however, that if the minimum width needed to display the contents of an actual column is greater than the width of a virtual column, then the width of the column is expanded to fit its contents.

- **HSPACE="horizontalMargin"**
  Specifies the distance between the left and right edges of the table and any surrounding content.

- **VSPACE="verticalMargin"**
  Specifies the distance between the top and bottom edges of the table and any surrounding content.

Similarly you can define Attributes within <TH> ... </TH> or <TD> ... </TD> cells as---

- ALIGN (LEFT, CENTER, RIGHT) Horizontal alignment of a cell.
- VALIGN (TOP, MIDDLE, BOTTOM) Vertical alignment of a cell.
- COLSPAN=n The number (n) of columns a cell spans.
- ROWSPAN=n The number (n) of rows a cell spans.

Important Hint

1. TH, TD and TR should always have end tags. Although the end tags are formally optional, many browsers will mess up the formatting of the table if you omit the end tags. In particular, you should *always* use end tags if you have a TABLE within a html document.

2. A default TABLE has no borders. By default, tables are drawn without border lines. You need the BORDER attribute to draw the lines.
3. By default, a table is flush with the left margin. Most current browsers also supports table alignment, using the ALIGN attribute. Allowed values are "left", "right", or "center", for example: <TABLE ALIGN="left">. The values "left" and "right" float the table to the left or right of the page, with text flow allowed around the table.
4. Alignment attributes on TD or TH attributes override those alignments specified by the TR:

➢ VARIOUS TABLE EXAMPLES

1.Table with no border and caption:

```
<HTML>
  <BODY>
      <TABLE>
         <CAPTION>TABLE WITH NO BORDER</CAPTION>
              <TR>
                <TD>ABC</TD>
               <TD>XYZ</TD>
               <TD>PQR</TD>
              </TR>
              <TR>
                <TD>LMN</TD>
                <TD>DEF</TD>
                <TD>HYJ</TD>
              </TR>
       </TABLE>
    <BODY>
  <HTML>
```

Output displayed by browser is:

```
TABLE WITH NO BORDER
                     ABC  XYZ PQR
                     LMN  DEF  HYJ
```

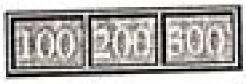
2.Table with background color and image no border:

```
<HTML>
 <BODY>
    <H4>A BACKGROUND COLOR:</H4>
        <TABLE BORDER="1" BGCOLOR="BLUE">
            <TR>
              <TD>100</TD>
              <TD>200</TD>
```

```
                <TD>300</TD>
              </TR>
        <H4>A BACKGROUND IMAGE:</H4>
            <TABLE BORDER="1"
      BACKGROUND=".C:\DESKTOP\XYZ.JPG">
               <TR>
                 <TD>SUNDAY</TD>
                 <TD>MONDAY</TD>
               </TR>
               <TR>
                 <TD>TUESDAY</TD>
                 <TD>WEDNESSDAY<TD>
               </TR>
             </TABLE>
      </BODY>
  </HTML>
```

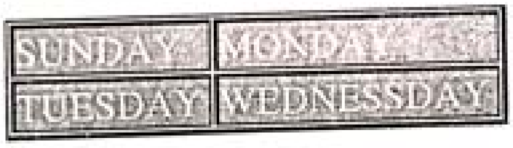
output displayed by browser is:

A Background Color:



100	200	300

A Background Image:



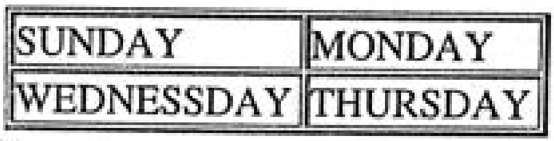
SUNDAY	MONDAY
TUESDAY	WEDNESSDAY

3.Table With Cell spacing

```
    <HTML>
      <BODY>
        <H4>WITHOUT CELLSPACING:</H4>
            <TABLE BORDER="1">
              <TR>
                <TD>SUNDAY</TD>
                <TD>MONDAY</TD>
              </TR>
              <TR>
                <TD>WEDNESSDAY</TD>
                <TD>THURSDAY</TD>
              </TR>
            </TABLE>
         <H4> WITH CELLSPACING:</H4>
            <TABLE BORDER="1" CELLSPACING="10">
              <TR>
                <TD> SUNDAY</TD>
                <TD> MONDAY </TD>
                </TR>
                <TR>
                  <TD> WEDNESSDAY </TD>
                  <TD> THURSDAY </TD>
                </TR>
              </TABLE>
         </BODY>
    </HTML>
```

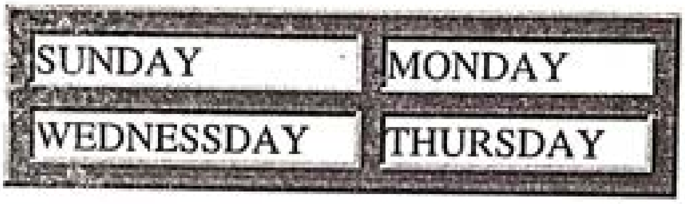
so output displayed by browser is:

WITHOUT CELL SPACING:



SUNDAY	MONDAY
WEDNESSDAY	THURSDAY

WITH CELL SPACING:



SUNDAY	MONDAY
WEDNESSDAY	THURSDAY

4.Table with cell padding

```
   <HTML>
     <BODY>
        <H4>WITHOUT CELLPADDING:</H4>
             <TABLE BORDER="1">
               <TR>
                  <TD>SUNDAY</TD>
                  <TD>MONDAY</TD>
               </TR>
               <TR>
                 <TD>WEDNESSDAY</TD>
                 <TD>THURSDAY</TD>
               </TR>
             </TABLE>
       <H4> WITH CELLSPADDING:</H4>
            <TABLE BORDER="1" CELLPADDING="20">
              <TR>
                <TD> SUNDAY </TD>
                <TD> MONDAY </TD>
              </TR>
              <TR>
                <TD> WEDNESSDAY </TD>
                <TD> THURSDAY </TD>
              </TR>
            </TABLE>
        </BODY>
    </HTML>
```

so output displayed by browser is:

WITHOUT CELL PADDING:

SUNDAY	MONDAY
WEDNESSDAY	THURSDAY

WITH CELL PADDING:

SUNDAY	MONDAY
WEDNESSDAY	THURSDAY

5. Table showing use of colspan and rowspan.

```
<HTML>
     <BODY>
        <H4>WITHOUT CELLPADDING:</H4>
             <TABLE BORDER="1">
               <TR>
                 <TD COLSPAN=2 ALIGN=CENTER>PRGRAMMING LANGUAGE</TD>
               </TR>
               <TR>
                 <TD ROWSPAN=2>OBJECT ORIENTED LANGUAGE</TD>
                 <TD>JAVA</TD>
               </TR>
               <TR>
                   <TD>C++</TD>
               </TR>
               <TR>
                   <TD ROWSPAN=2>STRUCTURED LANGUAGE</TD>
                   <TD>C</TD>
               </TR>
               <TR>
                   <TD>BASIC</TD>
               </TR>
            </TABLE>
        </BODY>
    </HTML>
```

so output displayed by browser. is :--

PRGRAMMING LANGUAGE	
OBJECT ORIENTED LANGUAGE	JAVA
	C++
STRUCTURED LANGUAGE	C
	BASIC

4.9 IMAGES IN HTML

The IMG tag specifies an image to be displayed in an HTML document.

An image can be a **plain image** that simply appears on the page. An image can be embedded in an <A HREF> tag so that the user can click it to open a URL. An image can also be an **image map**, which has **multiple clickable areas** that each link to different URLS. The **HEIGHT** and **WIDTH** attributes indicate the dimensions of the image. If you specify these attributes, Navigator uses them to reserve a place for the image on the page and continues loading any text and other page elements instead of waiting for the image to load. Most Web browsers can display inline images (that is, images next to text)

Images can be in the following types of formats:

- GIF (Graphics Interchange Format)
- JPEG (Joint Photographic Experts Group)
- XPM (X PixMap)
- XBM (X Bitmap)

Syntax

<IMG
SRC="*location*"
ALT="*alterntiveText*"
ALIGN="*alignment*"
BORDER="*borderWidth*"
HEIGHT="*height*"
WIDTH="*width*"
HSPACE="*horizMargin*"
VSPACE="*verticalMargin*"
>

The SRC attribute is compulsory

SRC="location(URL)"

specifies the URL of the image to be displayed in the document.

ALT="alternativeText"

specifies text to be displayed if the browser does not support the IMG tag or if the user has suspended image loading in the browser.

ALIGN

specifies the alignment of the image in relation to the surrounding text. If you do not specify a value for ALIGN, browser uses BOTTOM as the default LEFT aligns an image with the left margin.

- RIGHT aligns an image with the right margin.
- TOP aligns the top of an image with the top of the tallest item in the current line.
- TEXTTOP aligns the top of an image with the top of the tallest text in the current line.
- MIDDLE aligns the middle of the image with the baseline of the text in the current line.
- BASELINE aligns the bottom of an image with the baseline of the text in the current line.
- BOTTOM is the same as BASELINE.

BORDER="borderWidth"

Specifies the width, in pixels, of a border around the image. The value must be an integer.

HEIGHT="height"

Specifies the height of the image, either in pixels or as a percentage of the height of the window, frame, or positioned block of HTML that contains the image. To indicate a number of pixels specify the value as an integer, for example, "100". To indicate a percentage, specify the value as an integer followed by the percentage sign, for example "20%".

WIDTH="width"

Specifies the width of the image either in pixels or as a percentage of the window, frame, or positioned block of HTML containing the image. To indicate a number of pixels specify the value as an integer, for example, "100". To indicate a percentage, specify the value as an integer followed by the percentage sign, for example, "20%".

HSPACE="horizMargin"

Specifies a margin in pixels between the left and right edges of the image and surrounding text and images. Give the value as an integer

VSPACE="verticalMargin"

Specifies a margin in pixels between the top and bottom edges of the image and surrounding text and images. Give the value as an integer.

➢ Dummy Images

When an <IMG SRC> tag points to an image that does not exist, your browser software substitutes a dummy image. When this happens during your final review of your files, make sure that the referenced image does in fact exist, that the hyperlink has the correct information in the URL, and that the file permission is set appropriately.

➢ External Images, Sounds, and Animations

You may want to have an image open as a separate document when a user activates a link on either a word or a smaller, inline version of the image included in your document. This is called an external image, and it is useful if you do not wish to slow down the loading of the main document with large inline images.

To include a reference to an external image:

<A HREF="ExtImg.*gif*">text or link</A>

You can also use a smaller image as a link to a larger image :

<A HREF="*BigImage.gif*"><IMG SRC="*SmallImage.gif*"></A>

The reader sees the SmallImage.gif image and clicks on it to open the BigrImage.gif file.

You can use the same syntax for links to external animations and sounds. The only difference is the file extension of the linked file like:

<A HREF="K3GtheFilm.mov">link anchor</A>

specifies a link to a QuickTime movie. Some common file types and their extensions are:

1) plain text :-- . txt
2) HTML document :-- .html
3) GIF image :-- .gif
4) X Bitmap image :-- .xbm
5) JPEG image :-- .jpg or .jpeg
6) AIFF sound file :-- .aiff
7) AU sound file :-- .au
8) WAV sound file :-- .wav
9) QuickTime movie :-- .mov
10) MPEG movie :-- .mpeg or .mpg

- **Examples**
  - **Inserting Image as an background:**

Newer versions of Web browsers can load an image and use it as a background when displaying a page. Some people like background images and some don't. In general, if you want to include a background, make sure your text can be read easily when displayed on top of the image.

Background images can be a texture (linen finished paper, for example) or an image of an object (a logo possibly). You create the background image as you do any image.

```
<HTML>
  <BODY BACKGROUND="background Image.jpg">
  ..........text within a body
  </BODY>
</HTML>
```

Both gif and jpg files can be used as HTML backgrounds.
If the image is smaller than the page, the image will repeat itself.

- **Different sized images:**

```
<HTML>
    <BODY>
        <P>
            <IMG SRC="c:\MyDoc\bulb.gif" width="12" height="20">
        </P>
        <P>
            <IMG SRC="c:\MyDoc\bulb.gif" width="50" height="70">
        </P>
    </BODY>
  </HTML>
```

so output displayed by browser will be:

Thus you can make a picture larger or smaller changing the values in the HEIGHT" and "WIDTH" attributes of the img tag.

- **Images from different location:**

You can insert images from different locations either from web or from different directories on your local machine>Here you have to give the full path to src attribute, where image lies.

```
<HTML>
    <BODY>
        <P>
            Image from local disk,but different directory....
            <IMG SRC="c:\MyDoc\vk\images\bulb.gif>
        </P>
        <P>
          Image from net...
          <IMG SRC="http://www.yahoo.com/Logo.gif" WIDTH="73"
          HEIGHT="68">
       </P>
    </BODY>
</HTML>
```

- **Image as a link:**

You can have image as a link to some other image or document.

```
<HTML>
    <BODY>
        Click on Icon.gif file to get your destination document.
        <A HREF="DestinationPage.htm">
         <IMG BORDER="0" SRC="ICONT.GIF" WIDTH="20" HEIGHT="38">
        </A>
        <P>
         Click on Logo.gif file to get resultant image you want......
        <A HREF="ResultantImage.jpg">
         <IMG BORDER="0" SRC="ICONT.GIF" WIDTH="20" HEIGHT="38">
        </A>
    </BODY>
</HTML>
```

- **Alternate text:**

Text-only browsers will only display the text in the "alt" attribute, which is "This is Alternative Text".(in this example) or Some World Wide Web browsers such as Lynx (text only browser) cannot display images. Some users turn off image loading even if their software can display images (especially if they are using a modem or have a slow connection). HTML provides a mechanism to tell readers what they are missing on your pages if they can't load images The **ALT attribute** lets you specify text to be displayed instead of an image. Note that if you hold the mouse pointer over the image it will display the text called tool tip.

```
<HTML>
    <BODY>
       <IMG SRC="ICON.GIF" ALT=" This Is Alternative Text " WIDTH="30"
HEIGHT="30">
     </BODY>
</HTML>
```

- **Images without Text**

To display an image without any associated text (e.g., your organization's logo), make it a separate paragraph. Use the paragraph ALIGN= attribute to center the image or adjust it to the right side of the window

```
<HTML>
      <BODY>
         <P ALIGN=CENTER>
            <IMG SRC = "ICON.GIF" ALT="NOTEXT">
          </P>
       </BODY>
   </HTML>
```

4.10 HTML SCRIPTS

HTML pages with scripts are used to make them more dynamic and interactive.

➢ **VBScript**

Microsoft Visual Basic Scripting Edition brings active scripting to a wide variety of environments, including Web client scripting in Microsoft Internet Explorer and Web server scripting in Microsoft Internet Information Service.

Easy to Use and Learn

If you already know Visual Basic or Visual Basic for Applications (VBA), VBScript will be very familiar. Even if you do not know Visual Basic, once you learn VBScript, you are on your way to programming with the whole family of Visual Basic languages. VBScript is integrated with World Wide Web browsers. VBScript and Windows Script can also be used as a general scripting language in other applications.

VBScript talks to host applications using Windows Script. With Windows Script, browsers and other host applications do not require special integration code for each scripting component. Windows Script enables a host to compile scripts, obtain and call entry points, and manage the namespace available to the developer. With Windows Script, language vendors can create standard language run times for scripting. Microsoft will provide run-time support for VBScript

- **Insert a Script into HTML Page**

A script in HTML is defined with the <script> tag. Note that you will have to use the type attribute to specify the scripting language

e.g.

```
<HTML>
   <HEAD>
     <TITLE>JAVA SCRIPT</TITLE>
   </HEAD>
   <BODY>
     <SCRIPT TYPE="text/javascript">
           document.write("Hello World!")
     </SCRIPT>
  </BODY>
</HTML>
```

The script above will produce this output:

```
Hello World!
```

A new browser will understand that the script should be executed, even if comment tags surround it.

- **Various tags in VB script**
- **The <NOSCRIPT> Tag**

In addition to hiding the script inside a comment, you can also add a <noscript> tag.

The <noscript> tag is used to define an alternate text if a script is NOT executed. This tag is used for browsers that recognize the <script> tag, but does not support the script inside, so this browser will display the text inside the <noscript> tag instead. However, if a browser supports the script inside the <script> tag it will ignore the <noscript> tag.

e.g.

```
<SCRIPT TYPE="text/vbscript">
<!--
document.write("Hello World!")
-->
</SCRIPT>
<NOSCRIPT>Your Browser Does Not Support Vbscript!</NOSCRIPT>
```

- **Script Tags:**

<SCRIPT> :-- Defines A Script
<NOSCRIPT> :-- Defines An Alternate Text If The Script Is Not Executed
<OBJECT> :-- Defines An Embedded Object
<PARAM> :-- Defines Run-Time Settings (Parameters) For An Object
<APPLET> :-- Deprecated. Use <object>

The <SCRIPT> Tag

VBScript code is written within paired <SCRIPT> tags. For example, a procedure to test a delivery date might appear as follows:

```
<SCRIPT LANGUAGE="VBScript">
<!--
  Function CanDeliver(Dt)
    CanDeliver = (CDate(Dt) - Now()) > 2
  End Function
-->
</SCRIPT>
```

Beginning and ending <SCRIPT> tags surround the code. The LANGUAGE attribute indicates the scripting language. You must specify the language because browsers can use other scripting languages. Notice that the Can Deliver function is embedded in comment tags (<!— and —>). This prevents browsers that don't understand the <SCRIPT> tag from displaying the code.

You can use SCRIPT blocks **anywhere** in an HTML page. You can put them in **both the BODY and HEAD** sections. However, you will probably want to put all general-purpose scripting code in the HEAD section in order to keep all the code together. Keeping your code in the HEAD section ensures that all code is read and decoded before it is called from within the BODY section.

One notable exception to this rule is that you may want to provide inline scripting code within forms to respond to the events of objects in your form. For example, you can embed scripting code to respond to a button click in a form:

```
<HTML>
<HEAD>
<TITLE>Test Button Events</TITLE>
</HEAD>
<BODY>
<FORM NAME="Form1">
  <INPUT TYPE="Button" NAME="Button1" VALUE="Click">
  <SCRIPT FOR="Button1" EVENT="onClick" LANGUAGE="VBScript">
    MsgBox "Button Pressed!"
  </SCRIPT>
</FORM>
</BODY>
</HTML>
```

Most of your code will appear in either **Sub** or **Function** procedures and will be called only when specified by your code. However, you can write VBScript code outside procedures, but still within a SCRIPT block. This code is executed only once, when the HTML page loads. This allows you to initialize data or dynamically change the look of your Web page when it loads.

e.g Random number generator using VB Script **(Specimen paper)**

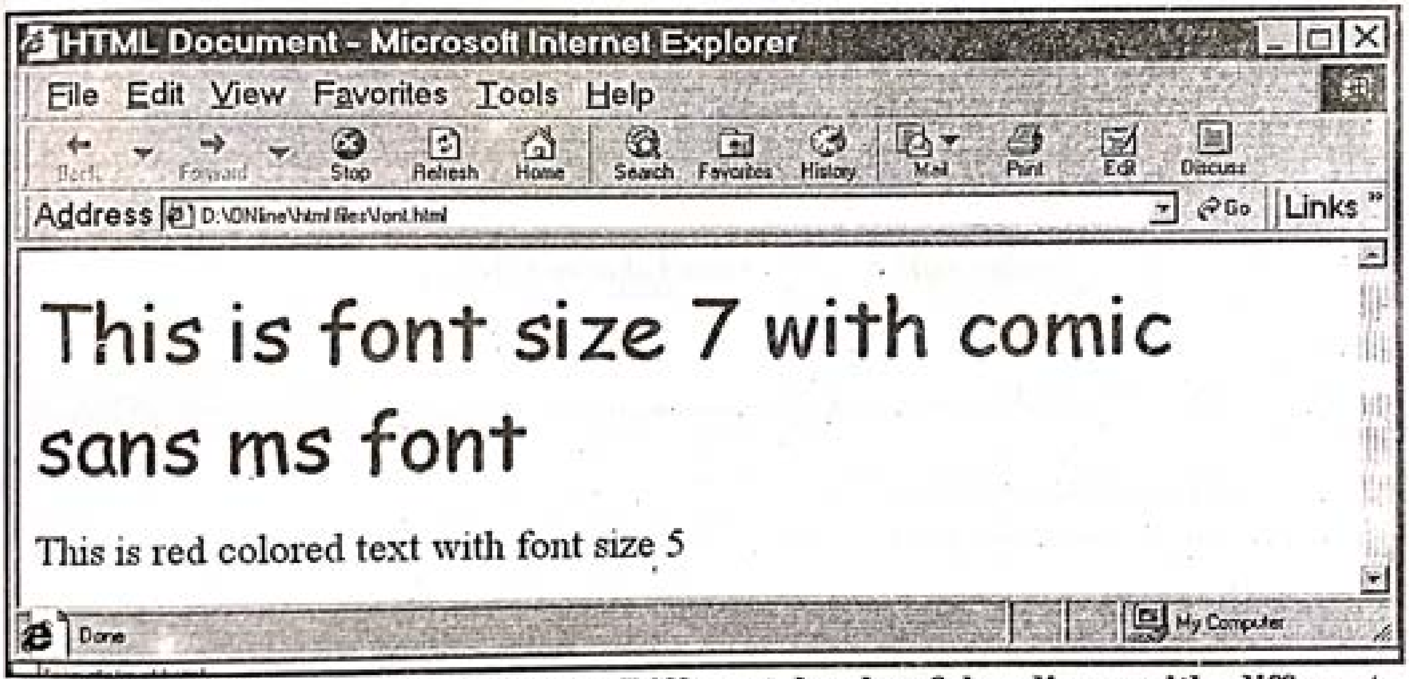
```
<%@ LANGUAGE="VBSCRIPT" %>
<%
 x=0
 y=0
 dim xarr(9)
 Do While x < 10
        Randomize
        Myval= Int((100 * Rnd) + 1)
        Y=0
        For y=0 to 9
                If MyVal = xarray(y) then
                        Match = True
                        Exit for
                Else
                        Match = False
                End If
        Next
                If Match = False then
                xarray(x) = MyVal
                x=x+1
                End If
        Loop
 %>
 <HTML>
        <BODY>
                <TITLE>RANDOM GENERATOR</TITLE>
        <%x=0'for x=0 to 9%>
       < P><%response.write(x+1)%>)&nbsp;<%response.write(xarray(x))%> </P>
        <%'next%>
        </BODY>
 </HTML>
```

SOLVED EXAMPLES

1) Write HTML code to display a Message using different Font attributes.

```
<HTML>
<HEAD>
<TITLE> HTML Document </TITLE>
</HEAD>
<BODY><FONT SIZE="7" face="Comic Sans MS">This is font size 7 with comic sans ms font</FONT>
<p><FONT SIZE="5" COLOR="red">This is red colored text with font size 5</FONT></p>
</BODY>
</HTML>
```

Output is:


HTML Document - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back
Forward
Stop
Refresh
Home
Search
Favorites
History
Mail
Print
Edit
Discuss
Address
D:\ONline\html files\font.html
Go
Links
Done
My Computer

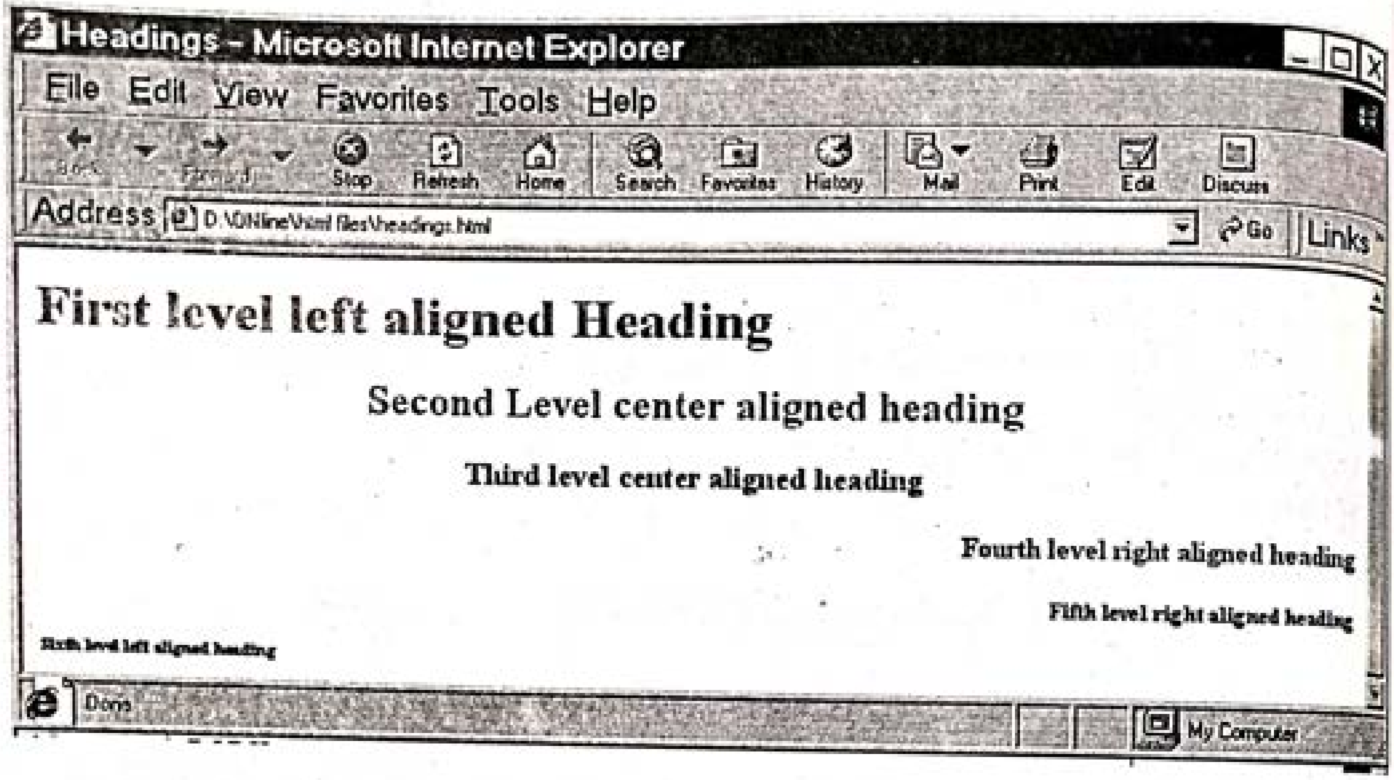
This is font size 7 with comic sans ms font

This is red colored text with font size 5

2) Write HTML code to explain Different levels of headings with different alignments.

```
<HTML>
<HEAD>
<TITLE>Headings </TITLE>
</HEAD>
<BODY>
<H1 align="left">First level left aligned  Heading</H1>
<H2 align="center">Second Level center aligned heading</H2>
<H3 align="center">Third level center aligned heading</H3>
<H4 align="right">Fourth level right aligned heading</H4>
<H5 align="right">Fifth level right aligned heading</H5>
<H6 align="left">Sixth level left aligned heading</H6>
</BODY>
</HTML>
```

Output is:


Headings - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Forward Stop Refresh Home Search Favorites History Mail Print Edit Discuss
Address
Go Links
Done
My Computer

First level left aligned Heading

Second Level center aligned heading

Third level center aligned heading

Fourth level right aligned heading

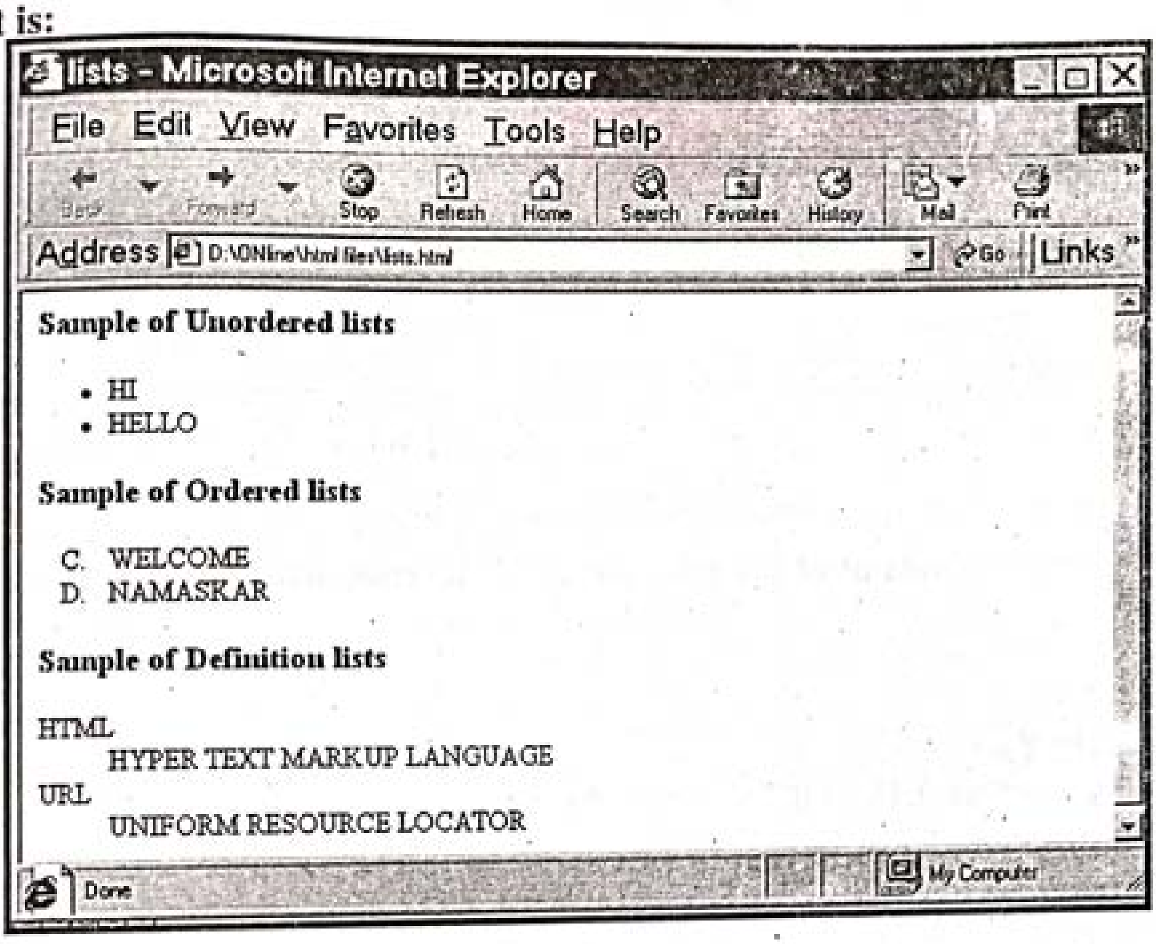
Fifth level right aligned heading

Sixth level left aligned heading

3) Write HTML code to explain Different Lists in Html.

```
<HTML>
<HEAD>
<TITLE> lists </TITLE>
</HEAD>
<BODY BGCOLOR="#FFFFFF">
<H3>Sample of Unordered lists</H3>
<UL type="disc">
        <LI>HI
        <LI>HELLO
</UL>
<H3>Sample of Ordered lists</H3>
<OL type="A" start=3>
        <LI>WELCOME
        <LI>NAMASKAR
</OL>
<H3>Sample of Definition lists</H3>
<DL>
        <DT>HTML
        <DD>HYPER TEXT MARKUP LANGUAGE
  <DT>URL
        <DD>UNIFORM RESOURCE LOCATOR
</DL></BODY>
</HTML>
```

Output is:


lists - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Forward Stop Refresh Home Search Favorites History Mail Print
Address D:\ONline\html files\lists.html
Go Links
Done
My Computer

Sample of Unordered lists

- HI
- HELLO

Sample of Ordered lists

C. WELCOME
D. NAMASKAR

Sample of Definition lists

HTML
HYPER TEXT MARKUP LANGUAGE
URL
UNIFORM RESOURCE LOCATOR

4) Write HTML code for the following: **(March 2002)**

<table>
<tr><th rowspan="2" colspan="2"></th><th colspan="3">YEAR</th></tr>
<tr><th>1998</th><th>1999</th><th>2000</th></tr>
<tr><th rowspan="2">SALES</th><th>UNITS</th><td>500</td><td>400</td><td>1000</td></tr>
<tr><th>INCOME</th><td>1000</td><td>800</td><td>2000</td></tr>
</table>

```
<HTML>
 <BODY>
  <TABLE BORDER>
       <TR>
              <TH ROWSPAN="2" COLSPAN="2">
              <TH COLSPAN="3">YEAR
              <TR>
                     <TH>1998
                     <TH>1999
                     <TH>2000
              </TR>
              <TR>
                     <TH ROWSPAN="2">SALES
                     <TH>UNITS
                            <TD>500
                            <TD>400
```

```
                        <TD>1000
            </TR>
            <TR>
                  <TH>INCOME
                        <TD>1000
                        <TD>800
                        <TD>2000
            </TR>
      </TABLE>
  </BODY>
</HTML>
```

5) Write the exact output of the following HTML code with font specifications i brackets: **(March 2002)**

```
<HTML>
   <BODY>
         <H1> LIST OF BOOKS </H1> <HR>
         <UL TYPE = "CIRCLE">
               <LI> How to solve it By computer
               <LI> HTML in Easy Steps
               <LI> C++ Programming
         </UL>
         <OL type = "A">
               <LI> Microprocessor Programming
               <LI> Networking Essentials
               <LI> Microcontrollers
         </OL>
   </BODY>
</HTML>
```

Output given by browser is:

List of books (text size hl)

- How to solve it By computer
- HTML in Easy steps
- C ++ Programming

A. Microprocessor Programming
B. Networking Essentials
C. Micro Controllers

QUESTIONS

Select the correct alternatives:

1) The____tag contains general information,or *meta*-information,about the document.

a) <HEAD> b) <MARQUEE> c) <TITLE> d)<BODY>

2) The block-level elements contain __________ tag.

a) <PRE> b) <BODY> c) <MARQUEE> d)<HTML>

3) __________ Attribute is used for setting color for visited hyperlinks

a) VLINK b)LINK c)HLINK

4) __________ specifies the distance between the left and right edges of the table and any surrounding content.

a) CELLSPACING b)CELLPADDING c)HSPACE d)VSPACE

5) A __________ tag, specifies the title for the table

a) CAPTION b)TITLE c)HEAD

6) In ____________type of list, list items are marked with bullets.

a) ORDERED b)DEFINITION c) NESTED d)UNORDERED

7) ________attribute in the body tag specify an image file to use as a background.

a) BGCOLOR b) BACKGROUND c)BGIMAGAE d) BACKGROUNDIMG

8) <a> tag has attribute ______which defines URL of the document to be linked.

a) SRC b) HREF c)VREF d)REF

9) Bulleted list in HTML is created by ________ tag. **(March 2019)**

a) <UL>, b) <OL>, c) <B>, d) <BR>

10) HREF attribute in <A>tag stands for__________ **(March 2002)**

a) Horizontal reference b)Hypertext reference

c)Hyperlink reference d)Hypermedia reference

11) To place the image into an HTML file ___ attribute is used in IMG tag.

a) < URL >, b) < ALT >, c), < SRC >, d) < HREF> **(March 2007)**

12) __________ tag is used to put a line break in HTML code. **(March 2003,2017)**

a)<HR> b) <BR> c) <P> d)<TR>

13) <A> tag has attribute ___ which defines URL of the document to be linked.

a) SRC b) HREF c) VREF d) REF **(March 2008)**

14) BORDER is attribute used in __________ tag of HTML. **(March 2016)**

a) < PRE >, b) < ADDRESS >, c), < TABLE >, d) < CAPTION>

15) _______ tag is used to create a row in table. **(March 2020)**

a) <td> b) <th> c) <tr> d) <tt>

2) What is HTML? State advantages and disadvantages of HTML. **(Oct.2003)**
3) What is ALT attribute? Why it is used?
4) Explain cellspacing, cellpadding, colspan and rowspan attributes.
5) How can you include image as an background and as an hyperlink? Explain giving examples?
6) With syntax diagram explain the structure of html webpage. **(March 2017)**
7) What are comments in HTML? Why they are required?
8) How can you have a link to a image by having image as a link?
9) Explain with example Nested lists in HTML.
10) Explain the following HTML tags with one example of each:
    i) <PRE>, ii) <SUP>, iii) <MARQUEE> **(Mar.2004)**
11) Explain the use of following HTML tags with example: **(Mar.2005)**
    i) <OL> ii) <TR> iii) <B>
12) Explain the following HTML tags with one example of each: **(Mar.2007)**
    i) <P> ii) <LI> iii) <PRE>
13) Explain the following HTML tags with one example of each: **(Mar.2008)**
    i) <MARQUEE> ii) <SUB> iii) <BODY>
14) What is character formatting in HTML?
15) Explain following tags with example:
    <sup> <marquee> <hr>
 <p> <script> **(March 2002)**
16) What are types of list? Explain with the example
17) What is script? Why it is required?
18) Explain changing color scheme of web pages.
19) Write advantages and disadvantages of HTML. **(March 2019)**
20) Explain general structure of HTML page. **(March 2020)**

➢ **Write HTML code to do following:-**

1.Write HTML code to do following.:--

a) Give title to your web page as "Exam".
b) Include background color as pink and text in green colour. Have a different font for page title and contents of the page.
c) Have a hyperlink to different page having url as "Result1.htm"
d) The resultant web page of this hyperlink should contain scrolling text as "All THE BEST " in bold, italic and black color and light blue as a background color.

2. Write HTML code to do following.:--

a) Give title to your web page as "TUTORIAL".
b) Include background color as light blue and
c) Text in red color.
d) Have a image "IMAGE.JPG" as an hyperlink.
e) The resultant web page of this hyperlink should contain scrolling text as"Best Of Luck" in bold, italic and green color. And "IMGNEW.JPG" image as the background of the page.

3.Write HTML code for following.:---

Collge Name:S.P.College.	
Admission data.	
Streams	*capacity*
Commerce	150
Science	100
Arts	100

Here Admission data. Should be the scrolling text.
Collge Name:S.P.College should be in bold, italics and in red color.
It should be hyper linked with th page containing college information.

4.Write HTML code for following.:---

Collge Name:Cummins College Of Engg.	
CCOEW	
Courses	No.of students.
Computer	120
Instrumentation	80
E & TC	60

Where image "LOGO.JPG" is shown as: CCOEW

5. Write HTML code to do following:

Todays Special:---

PIZZA (Domino's)	With toppings	Rs.150.	
	Without toppings	Rs.125.	
PavBhaji	Mashroom PavBhaji	Rs.40 .	
	Cheese PavBhaji	Rs.42. (WithButter)	Rs.40.
	Jain PavBhaji	Rs.40.	

6. Write the HTML code for the following table: **(Mar.2004)**

		Year		
		1999	**2000**	**2001**
Sales	**Units**	300	750	1,200
	Income	Rs. 3,000	Rs. 7,500	Rs. 12,000

7. Write a the HTML code for the following: **(Mar.2006)**

		Faculty		
		Arts	Science	Commerce
Students	Boys	100	400	500
	Girls	300	300	400

8. Write the exact output of the following HTML code with font specification in brackets: **(Mar.2006)**

```
<HTML>
        <TITLE> Introduction </TITLE>
        <BODY>
              <H1> <B> Computer Science </B></H1>
              <HR>
              <U> E Balaguru Samy </U>
              <HR>
              <H4> Achut S Godbole </H4>
        <BODY>
</HTML>
```

9. Write the HTML code for the following table. **(Mar.2008)**

		Year		
		2000	2001	2002
Sale	Units	500	1000	1500
	Income	Rs. 5,000	Rs. 10,000	Rs. 15,000

Answers Q. (1)

1) <HEAD>, 2) <BODY>, 3) VLINK, 4) HSPACE, 5) UNORDERED,

6) BACKGROUND, 7) HREF,8) Uniform Resource Locator,

9) <UL> 10))<BR> 11) < SRC >12) <BR> 13) HREF

14) <TABLE> 15) <tr>

□□□

PAPER-I
PRACTICALS

INTRODUCTION

The students offering this vocational Computer Science (D9) course are expected to be more perfect in practical. The course is more practical, and work experience oriented. The syllabus is arranged keeping in view that after completing this course, the candidate can write simple programs in assembly language, C++ and Visual Basic HTML code as well as execute the programs independently. This course would be definitely helpful with additional advantage to the students who will go for higher education i.e. Diploma and Degree course.

All practical for S.Y.J.C. Computer Science course are based on C++, HTML and VB for Paper-I and an assembly level language programming of 8085 for Paper-II. In this part of the book, guidelines for performing practical, scheme of examination, marks distribution, and other information are discussed.

(Note : For paper-II an instruction set with Opcode chart, ASCII chart are provided.)

Scheme of Examination

- The time duration for theory and practical examination is 3 Hours for each paper.
- During the examination, candidate has to perform only one experiment from those experiments, which have been conducted during the year.
- The term work 40 marks will be given by the internal teacher on the basis of performance of the student and the journal competed during the year.

HINTS FOR PRACTICAL (PAPER-I)

A) C++ Programming

1) Write down problem statement on your answer paper.
2) Try to analyse the problem and implement it in stepwise/algorithmic format. (need not draw algorithm/flowchart in answer paper)
3) Accordingly write C++ program.
4) Paper trace the program and try to find the output of it.
5) Code the C++ program in PC you can use VC++ editor or standard tcc (C++ IDE) editor
6) Save the file as filename .CPP
7) Include all the standard library files (e.g. string.h, io.h, etc) in your program.
8) Compile the program. If it contains any syntactical errors like missing semicolon, use keyword for declaring variables, etc., try to remove them.
9) After removing all the errors, run the program. If you are still not getting the required output, your program may contain logical errors. So use Debug utility for stepwise debugging of the program.
10) Finally run the program and take Hard copy of it.

B) HTML

1) Write HTML code in Notepad (or any standard text editor) if you are using windows operating system
2) Save that file as filename HTML
3) Open the same file through web browser loaded on your machine. (in Internet Explorer)
4) Output HTML file is interpreted by the browser
5) If you make changes in source code (HTML) file, and see the output for this change code, click on Refresh button in browser's window.

C) VISUAL BASIC

1) Read the problem statement and analyse visual design (i.e. which controls are to be used) and associated event procedures that forms coding part.
2) Using VB create a new project.
    a) First draw design-using controls on form and then write code in code window
    b) Run the project
3) Extension for form file is .frm and for project .vbp
4) During run mode you may get some errors. Correct it and execute again use watch facility.

Some Common Errors:

1) Try to avoid using keywords for variable names.
2) Try to match the brackets of the loops.
3) Indentations in a program helps you to figure out the mistakes in mismatching of { } brackets in a loop; so indent your program.
4) Declaration of all the variables that you use in your program is must.
5) Give necessary comments (especially for complex logic) whenever necessary. After successful running of program you can remove them.
6) If you use standard library functions like clrscr(), getch(), by seeing the help (ctrl+F1in C++ IDE editor) include required header files in your program.
7) In case of HTML indent your code. Give tags whenever necessary.
8) The order of nested tags should be proper.
9) If you have included image/document to be linked with some part of your document, give the whole path.
10) Try to avoid big/large-sized images, since it increases the size of memory.
11) To see the desired effect for a wave page, modify the source code and see the output interpreted through browser. Since browser cannot show you where error has occurred.
12) In VB If there are multiple forms in your project and if you want to execute only forms in that project. Always check startup object in project i.e. properties option.

C++ PROGRAMMING PRACTICALS

Slip No.1.SWAPING

Write a function in C++ that exchanges data (passing by reference) using swap function to interchange the given two numbers.

```
#include<iostream.h>
#include<conio.h>
//clrscr();
void swap(float &x,float &y)
{
     float t=x;
     x=y;
     y=t;

}
void main()
{void swap(float &,float &);
float a,b;
cin>>a>>b;
cout<<"a="<<a<<"b="<<b<<endl;
swap(a,b);
cout<<"a="<<a<<"b="<<b<<endl;
}
```

Output:

```
2
4
a=2b=4
x=4y=2
a=4b=2
Press any key to continue
```

Slip No.3 BINARY SEARCH

Write a program in C++ that first initialise an array of given 10-sorted real numbers. The program must verify whether a given element belongs this array or not, using **Binary Search technique**. The element (to be searched) is to be entered at the time of execution. If the number is found, the program should print its position in the array otherwise it should print, "The number is not found."

```
#include<iostream.h>
#include<conio.h>
#include<stdlib.h>
void main()
{
     float a[10],p;
int i,top,bot,mid;
cout<<"Type the number in ascending order"<<"\n";
for(i=0;i<10;i++){cin>>a[i];}
top=0;bot=9;
cout<<"Type the no you want to search \n";
cin>>p;
mid=(top+bot)/2;
while((top<=bot)&&(a[mid]!=p))
{

      if(p<a[mid])
            bot=mid-1;
      else
            top=mid+1;

      mid=(top+bot)/2;
}


      if(a[mid]==p)
      {
            cout<<"The number is at
position"<<(mid+1)<<"\n";

      }
      else
      cout<<"the number is not found\n";
}
```

Output

```
Type the number in ascending order
1
2
3
4
5
6
7
8
9
10
Type the no you want to search
7
The number is at position7
```

Slip No.7 RATIO AND RECIPROCAL

Write a program in C++ with a ratio class using member functions like assign ( ) function to initialize its member data integer numerator and denominator), convert( ) function to convert the ratio into double, invert ( ) function to get the inverse of the ratio and print ( ) function to print the ratio and its reciprocal.

```
#include<iostream.h>
class ratio
{public:
void assign(int,int);
double convert();
void invert();
void print();
private:
     int num ,den;
};
void main()
{ratio x;
x.assign(22,7);
cout<<"x=";
x.print();
cout<<"="<<x.convert()<<"\n";
x.invert();
cout<<"1/x=";x.print();
cout<<"\n";
}
void ratio::assign(int numerator,int denominator)
{
     num=numerator;
den=denominator;
}
double ratio::convert()
{
     return double((num)/den);
}
void ratio::invert()
{int temp=num;
num=den;
den=temp;
}
void ratio::print()
{cout<<num<<"/"<<den;
}
```

Output:

```
x=22/7=3
1/x=7/22
Press any key to continue
```

Slip No.8 CIRCLE AREA

Implement a circle class in C++. Each object of this will represent a circle, storing its radius and X and Y co-ordinates of its centers as floats. Include a default constructor, access functions, an area and circumference of the circle.

```
#include<iostreamo.h>
#include<conio.h>
class circle
{ private:
float x,y,r;
float area1,circum;
public:
            void assign();
      void area();
      void circumf();
      void print();
};
      void circle::assign()
      {cout<<"type the x & yco-ordinates of the centre
\n";
      cin>>x>>y;
      cout<<"type the radius";
      cin>>r;
      }
      void circle::area()
      {area1=3.14*r*r;}
      void circle::circumf()
      {
            circum=2*3.14*r;
      }
      void circle::print()
      {
            cout<<"x,y co-ordinate ="<<x<<"\t"<<y<<endl;
            cout<<"the radius is"<<r<<endl;
            cout<<"the area is"<<area1<<endl;
            cout<<"the circumference is"<<circum<<endl;
      }
      void main()
      {clrscr();
```

```
        circle c1;
        c1.assign();
        c1.area();
        c1.circumf();
        c1.print();
        }
```

Output:

```
type the x & y co-ordinates of the center
2
1
type the radius2
x,y co-ordinate =2      1
the radius is2
the area is12.56
the circumference is12.56  Press any key to continue
```

Slip No.9 OBJECT IS BORN

Write a program in C++ that initialises a Ratio class with no parameters, as a default constructor the program must print the message "OBJECT IS BORN" during initialisation. It should display the message "NOW X IS ALIVE", When the first member function Ratio x is called. The program must display "OBJECT DIES" when the class destructor is called for the object when it reached the end of its scope.

```
#include<iostream.h>
class ratio
{
public:
ratio()
{cout<<"object is born\n";}
~ratio()
{cout<<"object dies \n";}
private:
     int num,den;
};
void main()
{ratio x;
cout<<"now x is alive \n";
cout<< "at end of program \n";
}
```

Output:

```
object is born
now x is alive
at end of program
```

```
object dies
Press any key to continue
```

Slip No.10 COMPLEX NUMBER AND SUM

Write a program in C++ with a complex constructor to add the given two complex numbers. The program should print the given complex number and their sum.

```
#include<iostream.h>
class complex{
     float x,y;
public:
     complex(){}
     complex(float real,float img)
     {x=real;y=img;
     }
     complex operator+(complex);
     void display(void);
};
complex complex::operator +(complex c)
{complex t;
t.x=x+c.x;
t.y=y+c.y;
return(t);
};
void complex::display(void)
{
     cout<<x<<"+j"<<y<<"\n";
}

void main()
{complex c1,c2,c3;
c1=complex(2.5,3.5);
c2=complex(1.6,2.7);
c3=c1+c2;
cout<<"c1=";c1.display();
cout<<"c2=";c2.display();
cout<<"c3=";c3.display();
}
```

Output:

```
c1=2.5+j3.5
c2=1.6+j2.7
c3=4.1+j6.2
Press any key to continue
```

Slip No. 11 ADDITION AND DIVISION OPERATOR.

Write a program in C++ to implement the addition and division operator for the ratio class. Hence print the given two ratios x& y, their sum(x +y) and division(x/y);

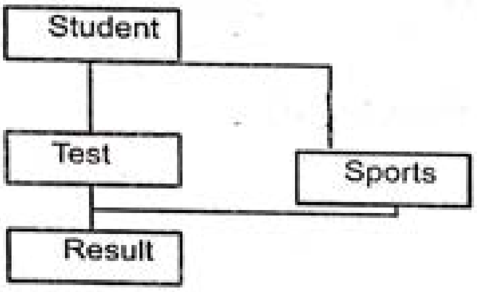
```
#include<iostream.h>
class ratio
{float x,y;
public:
     ratio(){}
     ratio(float p,float q)
     {x=p;
     y=q;
     }
     ratio operator + (ratio);
     ratio operator / (ratio);
     void display(void);
};
ratio ratio::operator +(ratio c)
{ratio t;
t.x=x+c.x;
t.y=y+c.y;
return(t);
}
ratio ratio::operator / (ratio c)
{ratio t;
t.x=x/c.x;
t.y=y/c.y;
return(t);
}

void ratio:: display(void)
{cout<<"x="<<x<<"\t"<<"y="<<y;
}

void main()
{ratio r1,r2,r3,r4;
r1=ratio(2.5,3.5);
r2=ratio(1.6,2.9);
r3=r1+r2;
r4=r1/r2;
cout<<"r1=";r1.display();cout<<"\n";
cout<<"r2=";r2.display();cout<<"\n";
cout<<"r3=";r3.display();cout<<"\n";
cout<<"r4=";r4.display();
}
```

Output:

```
r1=x=2.5      y=3.5
r2=x=1.6      y=2.9
r3=x=4.1      y=6.4
r4=x=1.5625    y=1.2069Press any key to continue
```

Slip No.12 CLASS HIERARCHY

a) Write a program in C++ to implement the following class hierarchy; Class student to obtain Roll Number, Class Test to obtain marks scored in two different subjects, Class sports to obtain weightage (marks) in sports and Class Result to calculate the marks. The program must print the roll number, individual marks obtained in two subject, sports and total marks.



```
#include<iostream.h>
class student
{
      int roll_no;
public:
      void get_no(int a){
            roll_no=a;
      }
      void put_no(void)
      {cout<<"roll_no:0"<<roll_no;
      }
};
class test:public student
{public:
      float s1,s2;
public:
      void get_marks(float x,float y)
      {s1=x;s2=y;
      }
      void put_marks(void)
      {cout<<"marks odtained : \ns1="<<s1<<"\n s2="<<s2;
```

```
     }
};
     class sports
     {
public:
     int score;
     void get_score(int x)
     {score=x;}
     void put_score(void)
     {cout<<"sports marks:"<<score;
     }
     };

     class result :public test,public sports
     {float tot;
     public :
          void display(void)
          {tot=s1+s2+score;
          put_no();
          put_marks();
          put_score();
          cout<<"total score="<<tot;
          }
     };
     void main()
     {result st1;
     st1.get_no(101);
     st1.get_marks(39,65);
     st1.get_score(29);
     st1.display();
     }
```

Output:

```
roll_no:0101marks obtained :
s1=39
s2=65sports marks:29total score=133
Press any key to continue
```

Slip No.13 VIRTUAL FUNCTION

Write a program in C++ using virtual function. The program must declare p to a pointer to objects of the base class person. First, the program must p to point an instance x (name of the person e.g. "BOB") of class person. The program must then assign p to point at an instance y (name of the student, e.g. "TOM") of the derived class student.

Define a print () function in the base class such that it invokes the same base class function to print the name of the person by default. The second call for the same should evoke the derived class the name of the student.

```
#include<iostream.h>
#include<string.h>
class person{
public:
     person(char* s)
     {
          name=new char[strlen(s+1)];
          strcpy(name,s);
     }
     void print()
     {
          cout<<"my name is"<<name;
     }
protected:
     char * name ;
};
     class student : public person{
          float rno;
     public:
          student(char *s,float r):person(s),rno(r)
          {}
          void print()
          {
               cout<<"my name is "<<name <<"my rno
is"<<rno;}
     };
          void main()
          {person *p;
          person x("bob");
          p=&x;
          p ->print();
          student y("tom",101);
          p=&y;
          p->print();
          }
```

Output:

```
     My name bob my name tom
```

Slip No.14 VISUAL BASIC

a) Write a program in VB to find the sum of first 100 numbers entered using DO Loop.

```
Rem *****************Do-Loop*************
        Private Sub Form_Load()
    Dim a, sum As Integer
    sum = 0
    a = 1
    Do
    sum = sum + a
    a = a + 1
    Loop While a <= 100
    MsgBox ("The sum of first 100 numbers is " & sum &
".") , vbInformation
    End

    End Sub
```

Slip No. 15 HTML

a) Create simple HTML pages on any of the following topics.

College profile, Computer Manufacturer, or Software Development Company.

The page must of at least 3 paragraphs of text. The page must have an appropriate title, background color or background image, and hyperlinks to other pages. The paragraphs must have text consisting of different colors and style in terms of alignment and Font size.

Save the file and view the same using any HTML Browser. Verify functioning of the hyperlinks.

b) Obtain a hardcopy of the HTML code only.

<u>Homepage:binary_soft.html</u>

```
 Source Code of Binarysoft.html
<html>
<head>
<h1 align="center">BINARY  SOLUTION</h1>
</head>
<body bgcolor="skyblue">
<p>
<font size=5 color="red">
<b>Binary Solution</b> is a software devlopment
company.It focuses on  Website devlopment.
It provide software solutions using Web and Dot Net
Technology.
```

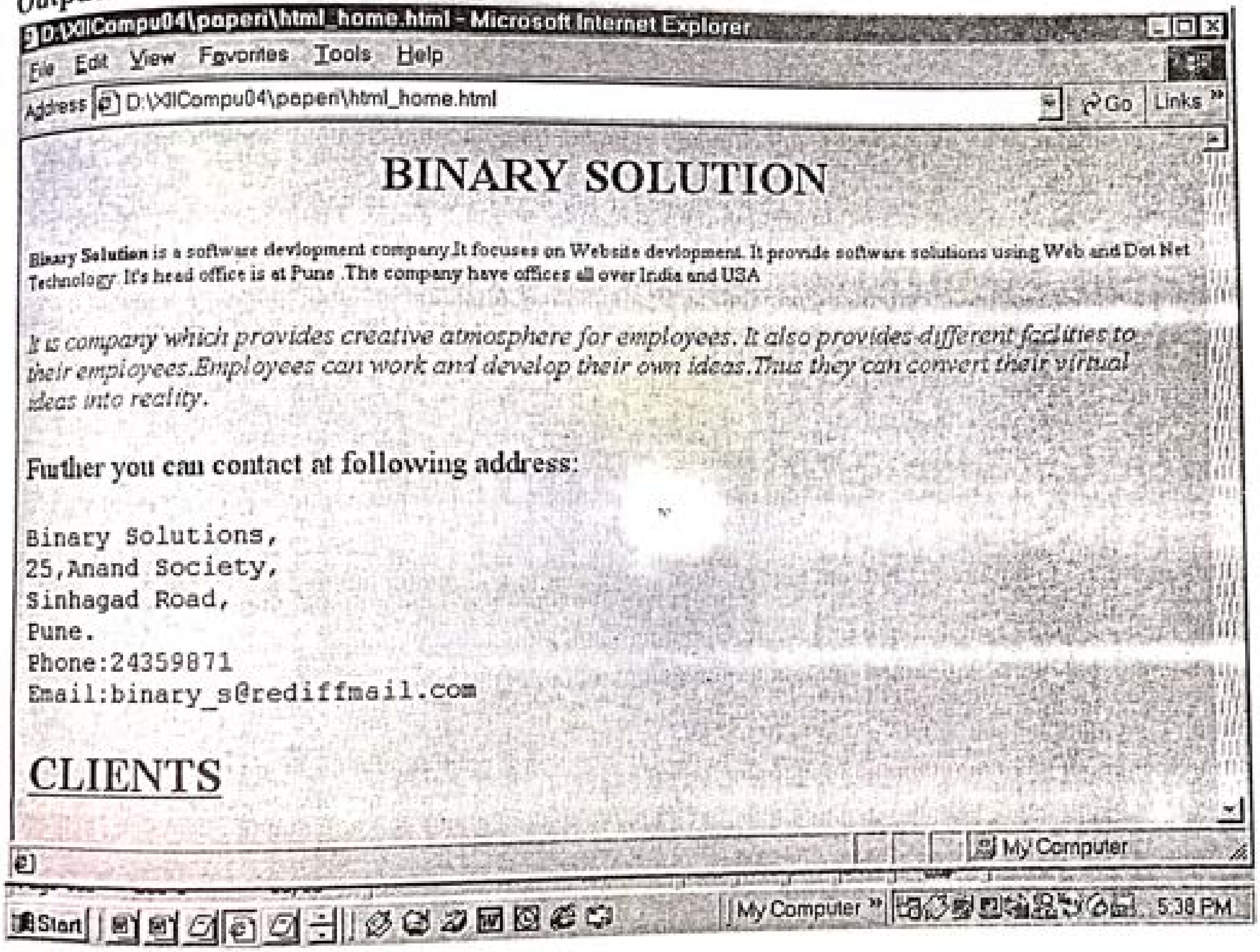
```
It's head office is at Pune .The company have offices all
over India and USA
</font>
</p>

<p>
<font size=+2 color="green">
<i>It is company which provides creative atmosphere for
employees.
It  also provides different faclities to  their
employees.Employees can
work and develop their own ideas.Thus they can convert
their virtual ideas
into reality.</i>
</font>
</p>

Further you can contact at following address:
<pre>
Binary Solutions,
25,Anand Society,
Sinhagad Road,
Pune.
Phone:24359871
Email:binary_s@rediffmail.com
</pre>
<A HREF="c:\taraka\clients.html">
<align="right"><h2>CLIENTS</h2></align> </A>

</body>
</html>
```

Link Page: clients.html

```
<html>
<head>
<h1 align="center">BINARY SOLUTION</h1>
</head>
<body bgcolor="skyblue">
<h1>Clients</h1>
<ol>
<li>Borgaon Sugar Factory</li>
<li>Indo Instruments</li>
<li>Nehru English School</li>
<li>ABC Publications</li>
```

```
<li>Indian Insurance Company</li>
<li>Toyato Limited</li>
</ol>
</body>
</html>
```

Output:


D:\XIICompu04\paperi\html_home.html - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Address D:\XIICompu04\paperi\html_home.html
Go Links
My Computer
Start
My Computer
5:38 PM

BINARY SOLUTION

Binary Solution is a software devlopment company.It focuses on Website devlopment. It provide software solutions using Web and Dot Net Technology. It's head office is at Pune .The company have offices all over India and USA

It is company which provides creative atmosphere for employees. It also provides different facilities to their employees.Employees can work and develop their own ideas.Thus they can convert their virtual ideas into reality.

Further you can contact at following address:

```
Binary Solutions,
25,Anand Society,
Sinhagad Road,
Pune.
Phone:24359871
Email:binary_s@rediffmail.com
```

CLIENTS

❑❑❑

COMPUTER HARDWARE

(PAPER-II)

1

MICROPROCESSOR AND ORGANISATION OF 8085

INTRODUCTION

Formerly digital computers were large expensive machines, now size and shape of these machines has been changed in past few years due to this new device "Microprocessor" also named as µp (mu-p). The microprocessor is an IC that contains much of the processing capabilities of a computer, it is small but extremely complex LSI device that is programmable. There are many companies which are manufacturing this microprocessor chips with different technologies and with different word length. Most popular companies in the manufacturing of µp are INTEL, MOTOROLA, Zilog, Toshiba. This chapter describes general features of a microprocessor, evolution of microprocessor and detail study of a very popular microprocessor chip Intel 8085-A as a basic programmable model of microprocessor.

1.1 EVOLUTION OF MICROPROCESSOR

In less than fifteen years we have seen five generations of microprocessors. Before the year 2000 AD a number of microprocessor in use will exceed the population of people on earth. The microprocessor is a twenty-year-old child of the integrated circuit and digital computer.

First Generation:(4-bit Microprocessor)

The first microprocessor introduced in the market was INTEL 4004, a 4-bit PMOS microprocessor, in 1971 by designed by scientist Faggin. This µp was in fact designed to be used in calculators. It was not powerful and was inadequate for general purpose computing.

In 1972 INTEL introduced the 8008, the first general purpose 8-bit µp with the development of LSI technology with 45 instructions. Intel introduced its successor, i.e. 8080. Simultaneously, Intel's competitors, inspired by the early design of 8080, introduced new µps. MOTOROLA introduced the 6800.

Second Generation : (8-bit Microprocessor)

In 1974 the successors to the 8080 and 6800 were introduced the Z80, 8085 from INTEL, 6809 from MOTOROLA etc. In 1976, 8085 was introduced. These µps were 8- bit microprocessors.

The development of µps has been in the direction towards a complete micro computer system with CPU, ROM, RAM, clock, I/O ports, all in single package. e.g. INTEL 8048, 8028, MOTOROLA MC 6801 etc.

1.2 ADVANCEMENT OF MICROPROCESSOR

Generation	Year	µP	Word Size in bits	General Information
First	1971	INTEL's 4004	4	First microprocessor 4 bit PMOS. It was designed to be used in calculators. It could perform arithmetic and logical operations.
		INTEL's 4040	4	Enhanced version of 4004
		Toshiba's T 3472	4	Similar to 4040
	1972	INTEL's 8008	8	First 8-bit µp. It could perform arithmetic and logic operations.
Second	1974	INTEL's 8080	8	Enhanced version of 8008. Showed developed in LSI.
		M6800 (Motorola) Z80 (Zilog) and	8 8	Second generation µps which we use commonly.
	1976	8085 (Intel)	8	µps developed in a direction to a complete microcomputer with CPU, Clock, I/O port all in a single IC package.
	1977	IM6100 (Intersil) T8190 (Toshiba)	12 12	12 bit µps developed as a enhancement in 8 bit µps.
Third	1978 to 1980	8086 (Intel) Zilog 8000 M 68000 Intel 8088	16 16 16 16	12 bit µps were followed by 16 bit µps. It showed a flow of progress of µps towards the CPU which has a capability to work with words bits, bytes.
Fourth	1981 1982	Intel 80386 iAPX 432 Hewlett Packard's HP32 M 68020	32 32 32	Latest advancement in µps. Intel specimen can address a physical memory of 4 Giga bytes.
		80486	32	Upgraded version of 386.
Fifth	1993 After 1996	Intel Pentium Intel Pentium II Intel Pentium III Celeron Xenon Dx4 i3 i7	64 64 64	Very popular in currently used PC's. Uses 32 bit address bus and speed of 64 MHz. to 233 MHz. In twentieth century, Intel introduced a series of advance processor for mobile, servers, internet with Dual core facility.

Table (1.1)

Third Generation : (16-bit Microprocessor)

The other major direction of µp evolution has been towards one which performs all functions of a minicomputer, which can work with words, bytes, strings of characters and instruction cycle of 1 microsecond. In 1978 INTEL introduced its high performance 16-bit microprocessor the 8086 now called as iAPX 86. It was introduced by Stephen Mors, Drus Ravenal and other scientist. This third generation memory space was 64 Kilo Bytes and number of I/O ports grew from 256 each in the 8080 to 64 K each in the 8086. Other features included efficient higher level language addressing, full arithmetic execution in 29 K transistors. The 8086 was followed by Zilog's Z8000 in 1979, Motorola's 68000 in 1980. e.g. INTEL 8086, 8088 , Zilog Z8000, MOTOROLA 68000 etc. All of these are 16-bit processors.

Fourth Generation: (32-bit Microprocessor)

In 1981, INTEL has introduced a first 32-bit microprocessor. The 80386, which can address a physical memory of 4 Gbytes (4 X 230G bytes). Other 32-bit µp announced 1982 as HP 32.

Fifth Generation: (64-bit Microprocessor)

Intel made a drastic improvement in microprocessor design to provide greatest speed and to run the system on new OS like UNIX. The processor in this generation is generally called by name Pentium, may be called as 80586 followed by PentiumII, III, IV etc.

Now a days after in twentieth century, Intel has developed so many processors with additional facilities like low voltage operating capability, processors handling mobile techniques, servers with Celeron, Xenon,i3,i7 dual core processors. Intel also produced Quad core processors with greater speed and additional features.

1.3 MICROCOMPUTER ORGANISATION

Computer system always contains five main blocks as (i) Input Device (ii) & (iii) Central Processing Unit with Arithmetic unit and control unit combined in it known by name (CPU) (iv) memory unit and (v) output device.

Let us see in brief how computer operates a given program.

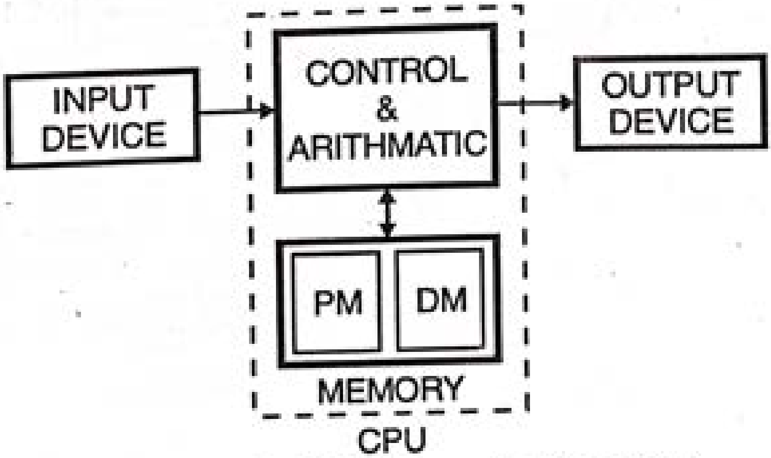


Fig. (1.1) Block Diagram of a Computer

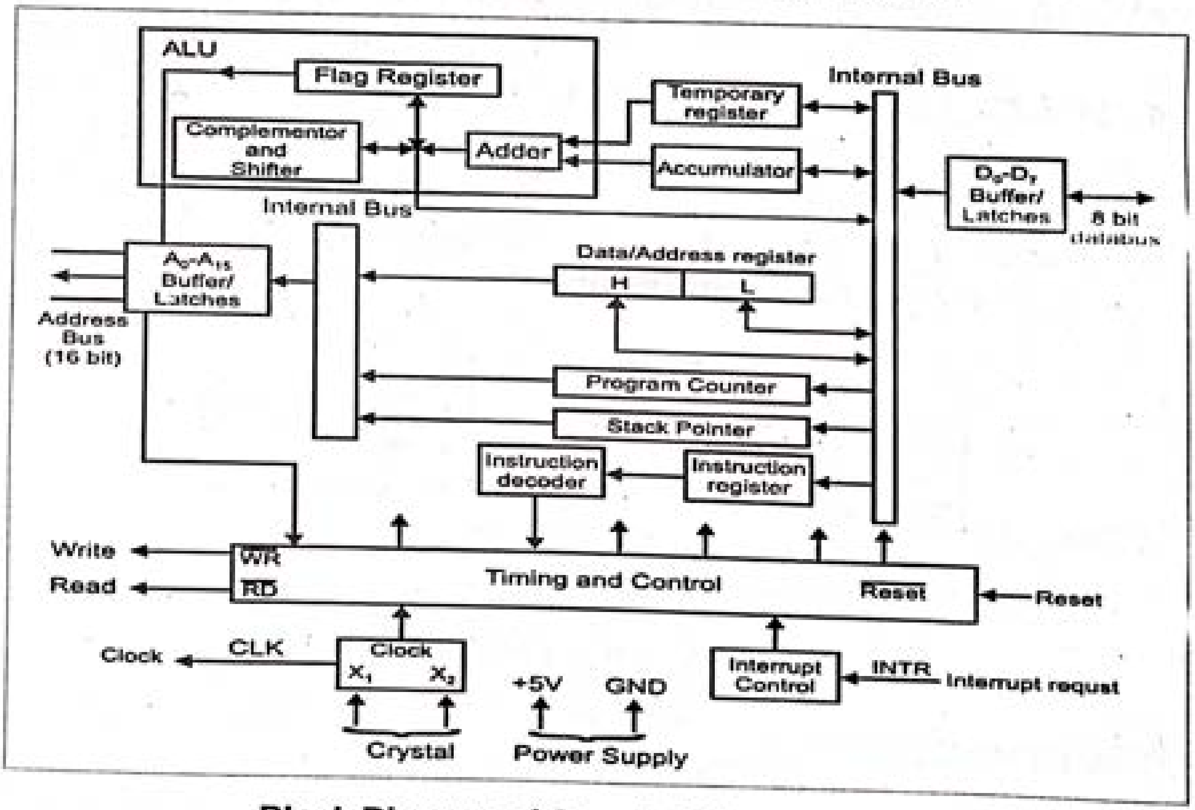
Fig (1.1) shows typical organisation of a computer with block diagram. A list of instructions with data is stored in a memory either for short time or permanently. Actually instructions are stored in a memory called as program memory (PM) and data is stored in other part of memory called as data memory (DM). After feeding set of instructions into the memory, control is used to read the first instruction from the program memory and according to the instruction it performs the work with the help of control and arithmetic circuit as adding of two numbers, jumping to other memory etc. Then after performing desired operation result is transferred to the output device. The basic difference between microprocessor and microcomputer is that microcomputer is a small size computer while microprocessor is one of the blocks required in a microcomputer system, in other way we can say microprocessor is a CPU part of the microcomputer system which actually is the heart of the system. Microprocessor with memory, input, output and clock form a microcomputer.

1.3 GENERIC MICROPROCESSOR

Generic Microprocessor gives an idea of simple architecture of a general microprocessor; it is a general model showing essential blocks in a Microprocessor. Figure shows a generic microprocessor with internal blocks connected in a logical structure to perform arithmetic and logical operations with data and address bus. To learn 8085; this model of generic microprocessor becomes helpful.

A generic microprocessor includes the following blocks and a required set of wires or lines called Bus:

1. Arithmetic logic unit (ALU)
2. Registers
3. Instruction decoder
4. Timing and control section.
5. Interrupt Control
6. Three types of bus


ALU
Flag Register
Complementor and Shifter
Adder
Internal Bus
Temporary register
Accumulator
Internal Bus
D_0-D_7 Buffer/ Latches
8 bit databus
A_0-A_{15} Buffer/ Latches
Address Bus (16 bit)
Data/Address register
H
L
Program Counter
Stack Pointer
Instruction decoder
Instruction register
Write
Read
$\overline{WR}$
$\overline{RD}$
Timing and Control
$\overline{Reset}$
Reset
Clock
CLK
Clock
X_1
X_2
Crystal
+5V
GND
Power Supply
Interrupt Control
INTR
Interrupt requst

Block Diagram of Generic Microprocessor

1. **Arithmetic logic unit (ALU)**

   It is an 8-bit unit where arithmetic and logical operations are carried out. After performing the operation it sets the flags according to the nature of resulting answer.

2. **Registers**

   Microprocessor makes use of several registers ;some of them are 16-bit and others 8-bit.

   i) Accumulator -8 Bit: Main register used to store Result of arithmetic and logical operation.

   General purpose registers

   ii) Register-B - 8 Bit
   iii)Register-C -8 Bit
   iv)Register-D - 8 Bit
   v)Register-E -8 Bit
   vi)Register-H - 8 Bit
   vii)Register-L -8 Bit
   viii)Temporary Register -8 Bit

   ix) Status register: it is as a set of flip-flops called as Flags. Microprocessor has two flags carry and zero flag.

   ix) Program Counter-16 bit: used to store the address of next memory location. It points or directs to the memory location containing the next instruction to be executed.

   x) Stack Pointer-16 bit used to store address of a memory called stack.

   xi) Instruction Register-8bit the first byte of instruction is loaded in this register.

3. **Instruction decoder**

   As its name suggests it takes instruction from Instruction Register and decodes it by driving control section.

4. **Timing and control section**

   Refer block diagram; as per signal given by instruction decoder it generates appropriate signals to control respective devices in order to execute the instruction.

5. **Interrupt Control**

   Interrupt is an essential process in microprocessor it executes certain steps when interrupt occurs.

6. **Three types of bus**

   Generic microprocessor shows three different types if parallel lines called bus.

   i) Address bus: it is 16-bit bus and unidirectional.

   ii) Data bus: it is 8-bit but bidirectional carries binary data.

   iii)Control bus: carries signal to execute operation.

1.5 BLOCK DIAGRAM OF MICROCOMPUTER

As explained in the above topic microcomputer contains all five blocks as (i) Input device (ii) Control (iii) Arithmetic circuit (iv) Program memory and data memory (v) Output device. Fig. (1.2) shows more detail architecture of a microcomputer in terms of block diagram with all essential bus and control signal lines.

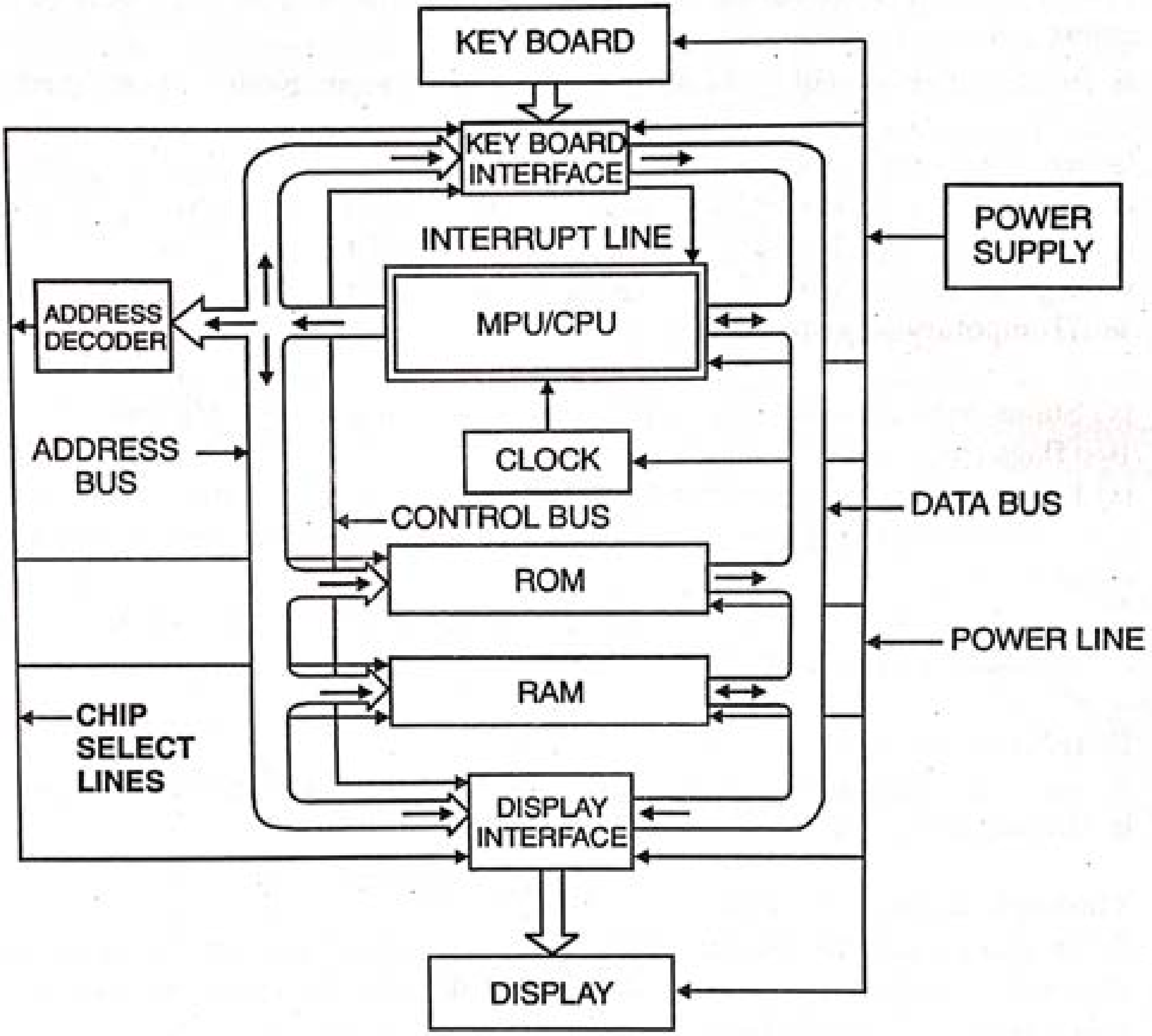

KEY BOARD
KEY BOARD INTERFACE
INTERRUPT LINE
POWER SUPPLY
ADDRESS DECODER
MPU/CPU
ADDRESS BUS
CLOCK
DATA BUS
CONTROL BUS
ROM
POWER LINE
RAM
CHIP SELECT LINES
DISPLAY INTERFACE
DISPLAY

Fig. (1.2) Block Diagram of a Microcomputer

As shown in the detailed architecture of a microcomputer it requires different bus structure, where bus is a group of conducting lines. It shows typical 16-bit *"address bus "*, there are 16 address lines and another bus is *"data bus"* as shown, it is 8-bit data bus depends on CPU. Address bus is normally unidirectional from the MPU (Micro Processor Unit) it sends an address of the required memory on this bus, while data bus is bi-directional from MPU data can be transferred to and from the MPU. The third group of bus is called as *"control bus"* through which control signals are carried. Control signals are generated by CPU and it is transferred on this bus to drive the required block of the system.

1. Input Device (Keyboard)

The instructions prepared for a particular program are entered through this device as shown in the fig.(1.2) through keyboard. At the same time data is entered with instructions. When instruction and data is entered through keyboard it is not possible to feed the instruction directly to MPU because MPU may be busy in performing previous instruction therefore it is stored in a keyboard interface. Another reason to connect keyboard input to keyboard interface is the speed of input device and MPU may not be equal. After storing the instruction and data it tells the CPU through a special signal line *"interrupt"* which interrupts the MPU. After MPU completes the current instruction it stops the normal execution and jumps to a special group of instructions in its monitor program, which gives the response to the keyboard data, is entered in the keyboard interface. Actually keyboard interface, is a special chip. Keyboard interface is connected to address bus, chip select and control lines, because when data is entered it is informed to the MPU through interrupt line therefore MPU generates the control signals and chip select signal to activate only this keyboard interface. MPU tells it to send the data on the data bus. Then MPU accepts the data and continues its current job.

2. Microprocessor Unit (MPU/CPU)

It can be referred as a CPU unit. In this main part of the system like brain of human being; data is processed and required control signals are generated to control the system. Whole processing and data flow is done with the MPU chip. Microprocessor is a controlling unit and fabricated on a very small chip capable of performing three main functions:

i) To receive information in digital form.
ii) To process this information by means of arithmetic and logical operations.
iii) To send the result or information in standard form.

3. ROM (Program Memory)

It contains the fixed stored program that is known as monitor program. It has address bus, chip select and read signal lines. ROM is read only memory. It allows only reading stored information.

4. RAM (Data Memory)

It is normally used to store data; it is a temporary storage device. Bi-directional data bus is required because to write and read the data into the memory. RAM has address input lines and chip select, read/write enable lines. RAM is normally 8-bit wide. It allows both read and write hence called read/write or Random Access Memory.

5. Output Device (Display)

Similar as input device and keyboard interface output of the system is displayed through display interface. The stored data in this display, interface is continuously transferred to 7-segment LED display.

6. **Address Decoder**

It samples the data and selects the proper device. Fig.(1.2) shows other important lines as (i) **Clock:** With which whole circuitry is synchronized. The speed of the system also depends on clock frequency. Refer block diagram, clock is given to MPU. (ii) **Power Lines:** The power supply is necessary to operate the circuits hence as shown all blocks are connected to this power supply line. Now let us see what is a microprocessor? How it executes the instruction with its organisation.

1.6 THE 8085 MICROPROCESSOR

The 8085A is a NMOS chip with 40 pin package and it is an 8-bit microprocessor suitable for a wide range of applications and it is ideal microprocessor to study because its basic architecture is used in more advanced microprocessors. The Intel's microprocessor 8085 chip has selected to understand hardware of the common microprocessor and to illustrate the assembly language program.

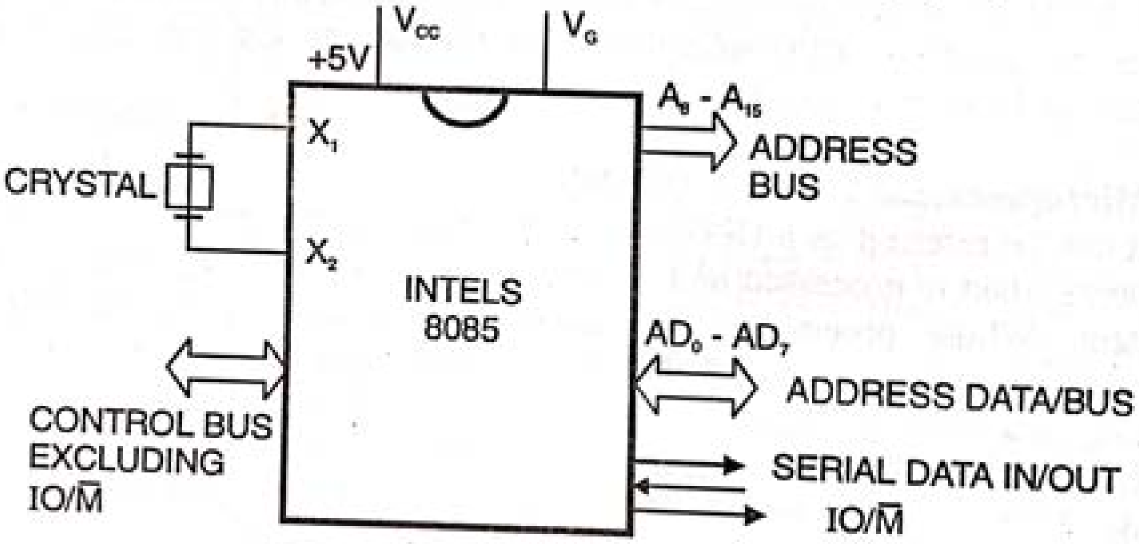


Fig. (1.3) Microprocessor 8085

FEATURES OF 8085

1. The Intel microprocessor 8085 is an 8-bit microprocessor 8-bit data bus width refer fig. (1.3). this indicates that 1 byte (8-bit) data can be transferred on this bus.
2. This 8085 chip is available in a 40 pin plastic ceramic (DIP) package.
3. The address bus is 16-bit that means it can address 64 K bytes. Refer fig. (1.3) and fig. (1.4) the address bus is divided into two groups as
   (a) The least significant bits of address are transmitted on the same eight lines of the data bus therefore it is called of data/address bus. This method of using common eight lines for both data and address transmission is known as *"multiplexed bus"*, (multiplexer means many into one here it is two into one).

   The advantage of multiplexed bus is no of pins are minimised from 16 to 8.

(b) Most significant bits of address are transmitted on address bus (A_8-A_{15}). Since microprocessor 8085 has 16 bit = (2^{16}) = 64 Kbytes memory locations can be accessed. Each memory location is of 1 byte that is 8-bit wide. In 8085 to select external memory or I/O device, I/O mapped, I/O system is used.

4. The instruction set of 8085 consists of 74 instructions.

1.7 FUNCTIONAL PIN DIAGRAM OF 8085

The pin diagram of 8085 is illustrated by fig. (1.4) and its brief functional description is given in table (1.2)

1) A_{D0}-A_{D7} - As shown in the pin diagram of 8085 has address/data multiplexed bus pins A_{D0}-A_{D7} and other address bus pins are A_8-A_{15}.

2) +Vcc and Vss - Supply connections are given across pins +Vcc and Vss ground +5V fixed.

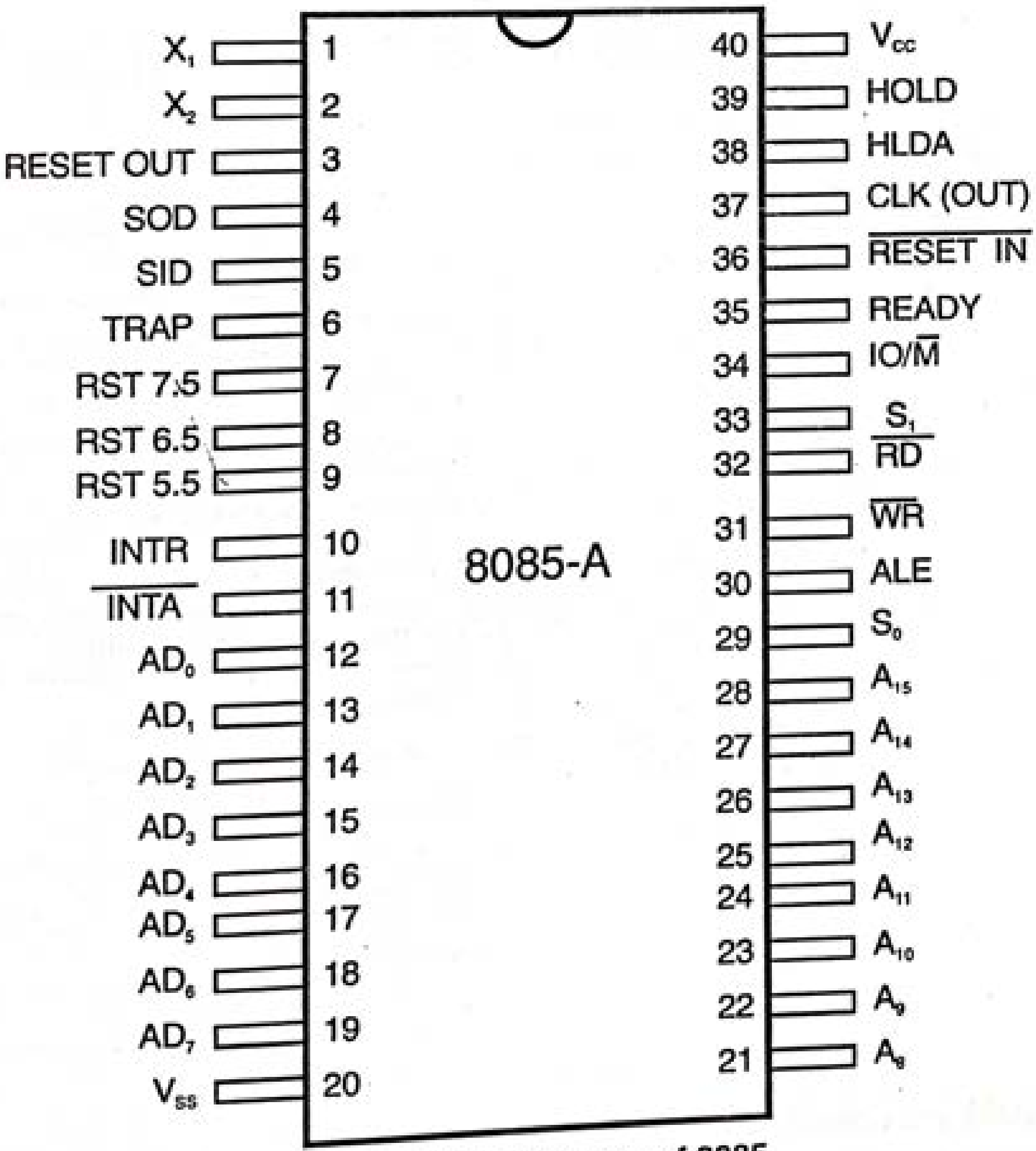

X1
X2
RESET OUT
SOD
SID
TRAP
RST 7.5
RST 6.5
RST 5.5
INTR
INTA
AD0
AD1
AD2
AD3
AD4
AD5
AD6
AD7
VSS
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
8085-A
40
39
38
37
36
35
34
33
32
31
30
29
28
27
26
25
24
23
22
21
VCC
HOLD
HLDA
CLK (OUT)
RESET IN
READY
IO/M
S1
RD
WR
ALE
S0
A15
A14
A13
A12
A11
A10
A9
A8

Fig. (1.4) Pin Diagram of 8085

Note: When Pin name is with Bar it is known as Active low Pin; it is activated with low voltage. Remaining pins without Bar activated with high level voltage.

Pin No.	Pin Name	Type	Description
1,2	X_1 X_2		Input Crystal Connections
3	RESET OUT	Output	Reset for Peripherals
4	SOD	"	Serial data output
5	SID	Input	Serial data can be input through this pin
6	TRAP	Input	Nonmaskable interrupt request
7	RST 7. 5	Input	Hardware Vectored Interrupt
8	RST 6.5	Input	--- do ---
9	RST 5.5	Input	--- do ---
10	INTR		Input Interrupt
11	$\overline{\text{INTA}}$	Output(Active low)	Interrupt acknowledged by 8085 on this line
12 to 19	$A_{DO} - A_{D7}$	Bi-directional Tristate	Address/Data bus
20	Vss	Input	Supply ground connection
21 to 28	A8 - A15	Output Tristate	Address bus MSB bits.
29	So	Output Tristate	Address bus MSB bits
30	ALE	Output	Address latch enable control
31	$\overline{\text{WR}}$	Output Tristate	Write control(Active low)
32	$\overline{\text{RD}}$	Output Tristate	Read control(Active low)
33	S1	Output Tristate	I/O or memory control
34	IO/$\overline{\text{M}}$	Output Tristate	I/O or memory selector control
35	READY	Input	Wait request
36	$\overline{\text{RESET IN}}$	Input (Active low)	System RESET active low pin
37	Clock (OUT)	Output	Clock signal
38	HLDA	Output	Hold is acknowledge
39	HOLD	Input	Request to hold
40	Vcc (+5V)	Input	Positive supply connection

Table (1.2)

3) X_1-X_2 - 8085 microprocessor has its own built in oscillator (Clock) circuit inside it to which externally crystal is connected across pins X_1 and X_2.

4) $\overline{RD}$ and $\overline{WR}$ – These are active low pins. A low level $\overline{RD}$ signal indicates the selected memory/IO is to be read and data bus is available for data transfer. Similarly, a low level $\overline{WR}$ signal indicates that data on data bus is to be written into the selected memory/IO location.

5) $\overline{\text{RESET IN}}$/RESET OUT – Pins are used to make program counter of 8085 reset to 0000H. After program counter is reset by peripheral through $\overline{\text{RESET IN}}$ pin, microprocessor sends signal through output pin RESET OUT that the "program counter is reset to inform peripheral.

6) Interrupt pins - Microprocessor 8085 has five interrupt-pins through which 8085 is interrupted (*Interrupt is explained in later topic).

7) SID/SOD - For serial data transmission SID and SOD pins are used. For this type of transmission RIM and SIM instructions are required.

8) READY - The Pin READY input from peripheral device informs the microprocessor that is peripheral device is ready to receive or to send the data.

9) HOLD/HLDA - HOLD input pin is required during direct memory access (DMA) operation that is HOLD signal informs microprocessor that another device requires to use address and data bus. After receiving the signal HOLD request has been received on the pin HLDA (holds acknowledge).

ADDRESSING I/O DEVICES (IO/$\overline{M}$)

When microprocessor is connected with other devices as memory and input-output devices it forms a microcomputer. Sometimes to select particular input device and output device particular address is given as a memory locations for example address 0002 is used to select input device like keyboard and 0003 for output device like printer. Therefore whenever address is given by, microprocessor 0002 or 0003 then there is no other information in these memory locations. In other words when these two addresses are used to select I/O devices then these two addresses are not used for any memory location. This type of I/O addressing is known as ***"Memory mapped I/0"***. But in microprocessor 8085 ***"I/O mapped I/O"*** method is used for I/O devices. For example the address 0002 and 0003 when it appears on address bus then this address may be memory or for I/O which depends on the control signal present at pin IO/ $\overline{M}$.

When IO/$\overline{M}$ = 1 then this address is for I/O device and when IO/$\overline{M}$ = 0 then it is for memory. In microprocessor actually IO/$\overline{M}$, S_1 and S_0 control signal decide the type of operation. It is given in table (1.3)

Refer table (1.3) when IO/$\overline{M}$ = 0 then memory is selected either read or write which is decided by S_0 and S_1 and when IO/$\overline{M}$ = 1 then I/O device is selected to read or write decided by S_0 and S_1 signal.

$IO/\overline{M}$	S_1	S_0	Operation of 8085
0	0	1	Memory write
0	1	0	Memory read
1	0	1	I/O write
1	1	0	I/O read
0	1	1	OP code fetch
1	1	1	Interrupt acknowledge
Floating	0	0	Halt
Floating	-	-	Hold
Floating	-	-	Reset

Table (1.3)

Let us see the difference between memory mapped I/O and I/O mapped I/O.

Memory mapped I/O	I/O mapped I/O.
1) I/O is assigned to the separate address space of memory.	1) I/O is assigned with same address of memory.
2) Separate control signal is not required.	3) It is required like $IO/\overline{M}$.
3) Memory space is reduced due to insertion of I/O address.	4) Full memory space can be used.
4) It requires 16-bit address bus.	5) It uses 8-bit address bus only.

1.8 INTERNAL BLOCK DIAGRAM OF 8085

Internal block diagram many times is referred as organisation or architecture of 8085. It is shown by fig. (1.5)

Figure (1.5) shows following main blocks as:

1) Three types of bus
2) Registers
3) Arithmetic logic unit (ALU)
4) Flags
5) Program Counter (PC)
6) Stack Pointer
7) Incrementer / Decrementer
8) Timing and control section.

Let us see the significance of each block in 8085

1. Three Bus Structure

Refer fig. (1.5) there are two I/O bus as 8-bit data bus and 16-bit address bus. As mentioned earlier $A_{D0} - A_{D7}$ bus is address/data-multiplexed bus. Control bus is not shown but different signals are transferred to different blocks like $\overline{WR}$, $\overline{RD}$ ALE etc, which are generated by control section that forms a control bus. It has three types of control bus refer fig. (1.5), the top side one for interrupt control which includes INTR, INTA, RST 5.5, RST 6.5, RST 7.5 and TRAP. The second one is for serial communication SID/SOD. The third control bus is the main control bus. It generates all control signals which are applied internally and externally to execute an instruction.

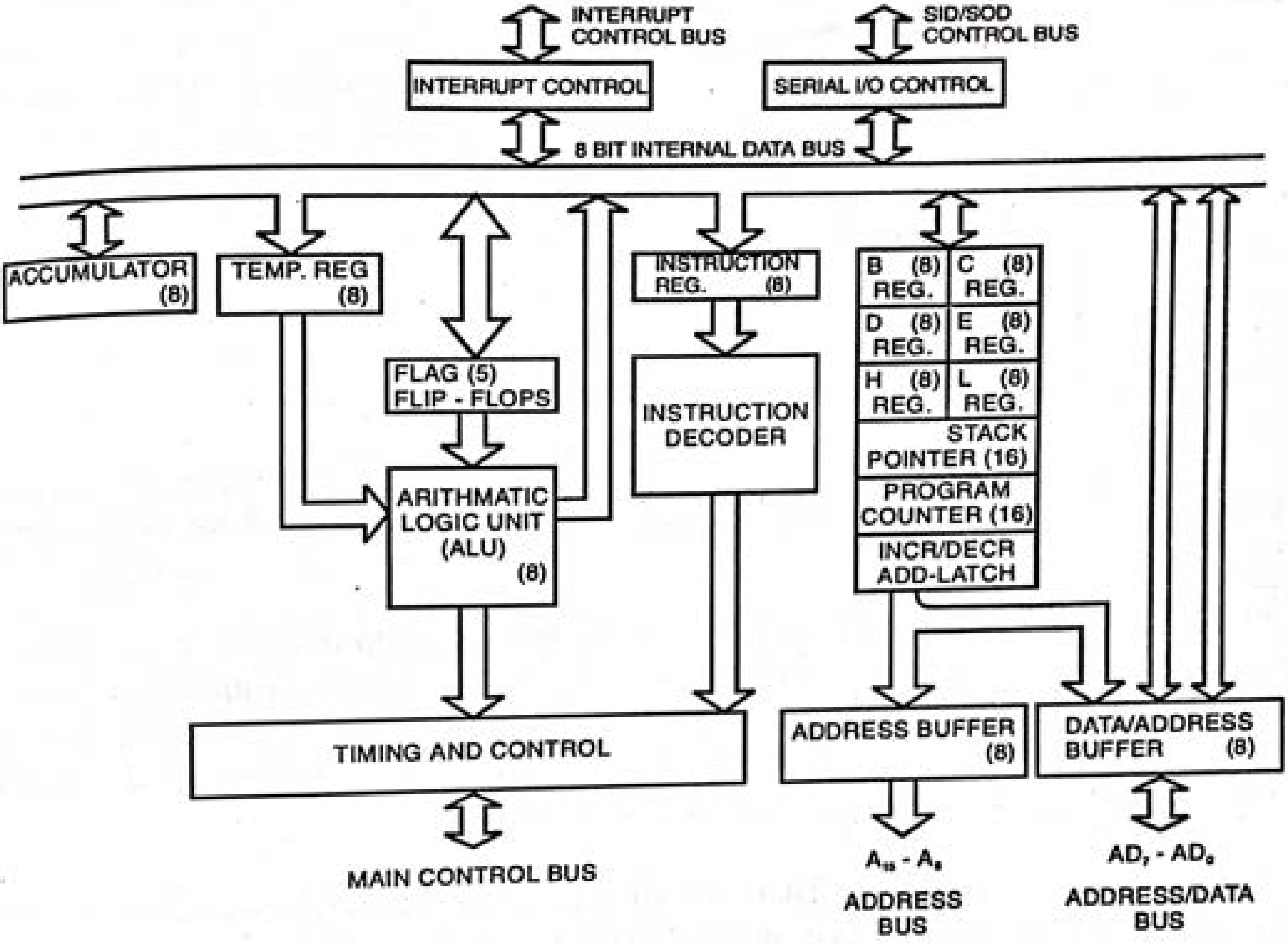

INTERRUPT CONTROL BUS
SID/SOD CONTROL BUS
INTERRUPT CONTROL
SERIAL I/O CONTROL
8 BIT INTERNAL DATA BUS
ACCUMULATOR (8)
TEMP. REG (8)
FLAG (5) FLIP - FLOPS
INSTRUCTION REG. (8)
INSTRUCTION DECODER
B (8) REG.
C (8) REG.
D (8) REG.
E (8) REG.
H (8) REG.
L (8) REG.
STACK POINTER (16)
PROGRAM COUNTER (16)
INCR/DECR ADD-LATCH
ARITHMATIC LOGIC UNIT (ALU) (8)
TIMING AND CONTROL
ADDRESS BUFFER (8)
DATA/ADDRESS BUFFER (8)
MAIN CONTROL BUS
A_{15} - A_8
ADDRESS BUS
AD_7 - AD_0
ADDRESS/DATA BUS

Fig. (1.5) Internal Block Diagram of 8085

2. **Registers**

Refer fig. (1.5), there are different registers used for different purposes as

A) 8-Bit General Purpose Registers

i) Accumulator - 8 Bit
ii) Register-B - 8 Bit
iii) Register-C - 8 Bit
iv) Register-D - 8 Bit
v) Register-E - 8 Bit
vi) Register-H - 8 Bit
vii) Register-L - 8 Bit
viii) Temporary Register - 8 Bit
ix) Flag/Status Register - 8 Bit

Note: Accumulator with flag Register is known as Program Status Word (PSW)

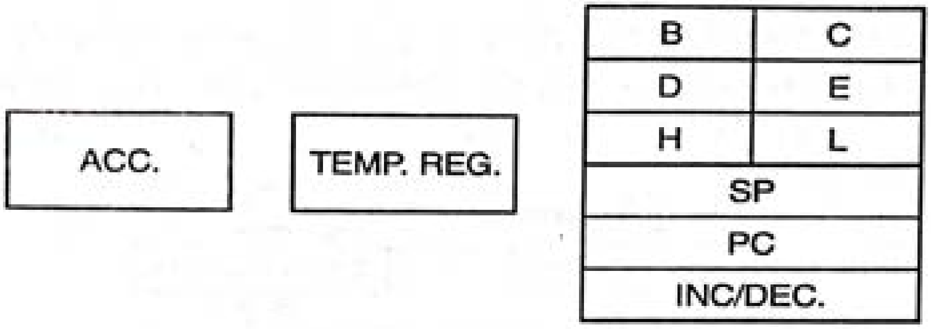


Fig. (1.6) Different Registers

B) 16-Bit Special Registers

i) Program Counter - 16 Bit
ii) Stack pointer - 16 Bit
iii) Incrementer–Decrementer - 16 Bit

Here register B,C, D, E, H and L can be used as 16-bit registers in pairs as register pair B-C, register pair D-E register pair H-L. Remember 16-bit register pair cannot be formed like B-E or D-L.

i) **Accumulator:** It is an 8-bit main register in 8085 microprocessor; it is used for performing arithmetical and logical operations like addition, subtraction, AND EX-OR operations. One of the operand is stored in accumulator. It can be used as both source and destination register. The final result of these operations is also stored in this accumulator. Accumulator also used for I/O operations.

ii) **General purpose registers:** There are six 8-bit registers which can be used as 16-bit registers in pairs as B-C, D-E and H-L pair. The higher MSB 8-bits are stored in first register of the pair as B,D,H and lower LSB 8-bits are stored in other register of C,E,L. Suppose a number 24C5H is to be stored in D-E registers pair then 24 number will be stored in register D and number C5 will be stored in register E.

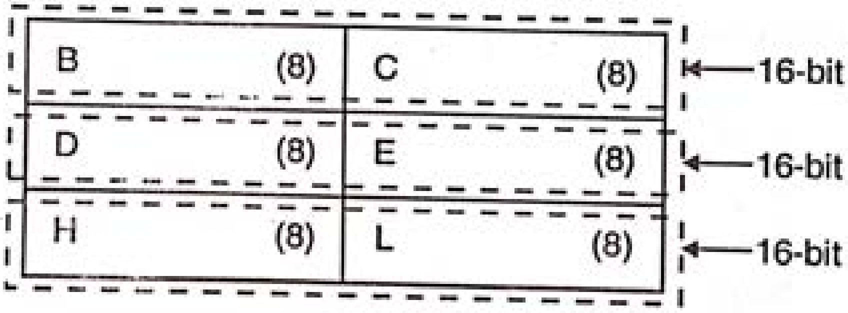


Fig. (1.7) General Purpose Registers

iii) **Temporary Register:** This temporary register is used to store operands of arithmetic and logical operations. Similar to accumulator other 16-bit registers program counter, stack pointer etc are used for special purpose as explained below.

3. ARITHMETICLOGIC UNIT (ALU)

It is an 8-bit unit where arithmetic and logical operations are carried out. The ALU consists of binary adder it performs only binary addition and subtraction by the 2's complement addition method. Accumulator, I/O device, memory etc, supplies data. After performing the operation it sets the flags according to the nature of resulting answer. If answer is zero then it sets zero (Z) flag. Detail explanation of flags is given below.

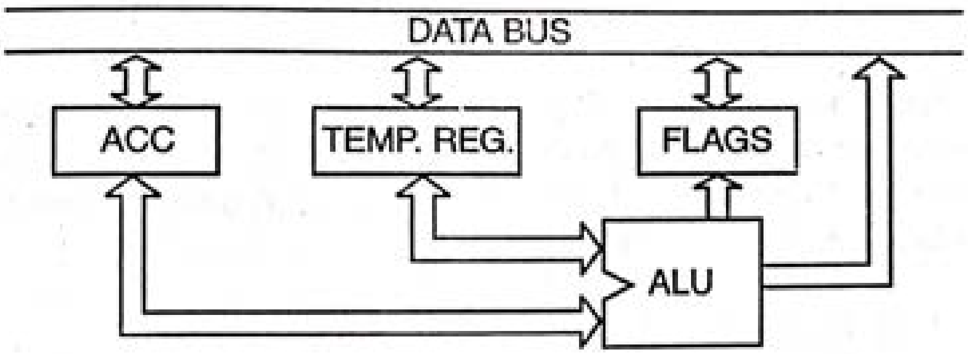


Fig. (1.8) ALU Circuit

4. FLAGS

These are single bit status registers (Flip-flops) operated by ALU. The flags are either set or reset according to the answer produced by ALU. It is important because flags are examined in conditional instructions like *"JMP" &"CALL"*.

There are five flags:

1) Sign Flag (S), 2) Zero Flag (Z), 3) Auxiliary Carry Flag (AC),
4) Parity Flag (P), 5) Carry Flag (CY),

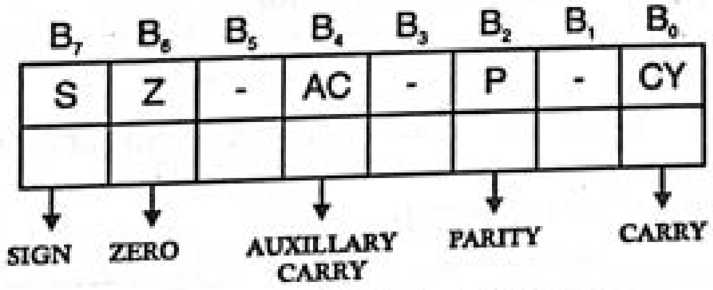


Fig. (1.9) Flag Register

Condition and Code		Flag 0 = Reset, 1 = Set
Not Zero	(NZ)	Z = 0
Zero	(Z)	Z = 1
No carry	(NC)	CY = 0
Carry present	(C)	CY = 1
Parity Odd	(PO)	P = 0
Parity Even	(PE)	P = 1
Plus	(P)	S = 0 and when D_7 bit of Acc= 0
Minus	(M)	S = 1 and when D_7 bit of Acc= 1

Table (1.3)

With instruction STC carry flag is directly set. Table (1.3) illustrates different conditions of ALU result and the status of flags affected.

After execution of arithmetic operation programs status word (PSW) is formed. It contains accumulator with flag.

4. PROGRAM COUNTER (PC)

Program counter is a 16-bit register used to store the address of next memory location. It points or directs to the memory location containing the next instruction to be executed. For example an instruction to the 8085 is suppose LXI B, 2475H this means load register pair with 2475H. Now 8085 should give address of instruction LXI B in to program counter. Suppose it is stored in memory location 0200H then program counter will get incremented by microprocessor to 0201 H, 0202H. Let us take a simple program as follows:

```
PC
0200   LXI  B,2475H          (1)
0201   LDAX  D               (2)
0202   LXI D, 3794H,         (3)
```

In this program our instruction is in first line. After completing instruction no. (1) program counter contents automatically goes to 0201H which is the address of next instruction LDAX D then it will go to 0202, 0203 .... automatically. Sometimes a JUMP instruction is present in the main program then normal sequence is disturbed. JUMP instruction directs some other (X) memory location therefore program counter will contain the address of (X) memory location. So that is provides logical continuity to the program. Thus microprocessor increments the program counter every time as it fetches the instruction.

5. STACK POINTER (SP)

It is a 16-bit register used to store an address of current memory called stack.

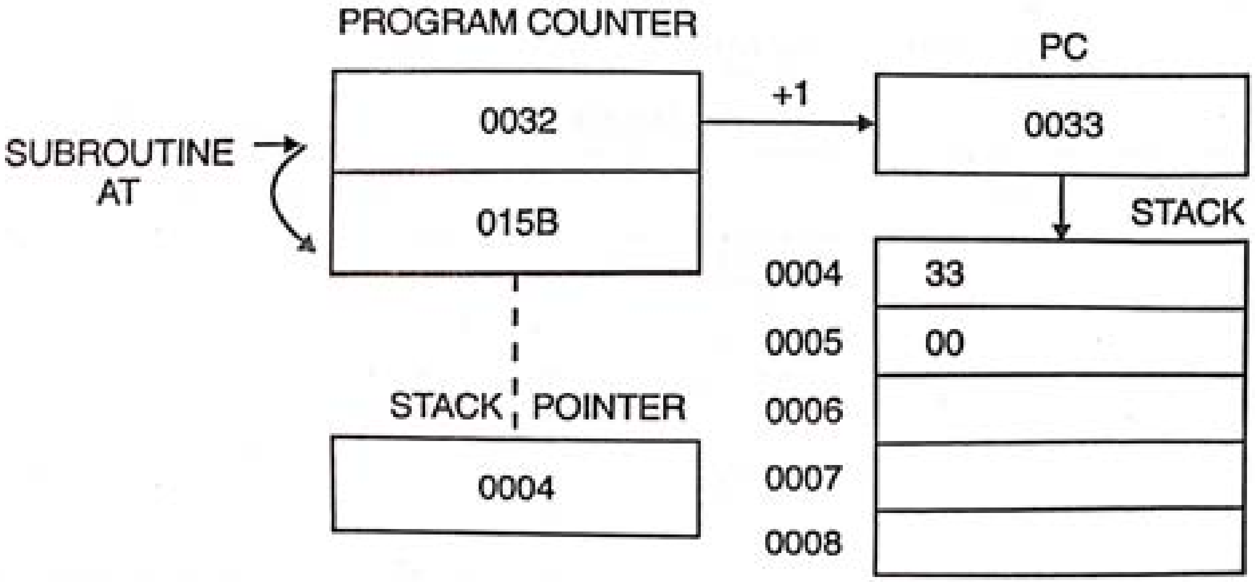


Fig. (1.10) Idea of Stack and Stack Pointer

As explained above, in case of subroutine the program counter is directed to the address of branching memory location. But before that program counter is incremented by 1, which will be the address of next instruction in the main program, the address is stored in this special group of register or memory area called as stack. The address of stack is initialized by LXI H instruction.

Refer Fig. (1.10) suppose program counters current address is 0032 and a subroutine occurs at this instruction. Therefore before program counter goes to next address of subroutine (as shown 015B) it will be incremented by 1 that is 0032 + 1 = 0033 this will be the next address, which is saved in stack. Stack is a part of RAM. Address 0033 is stored in stack at memory location of stack 0004 and 0005. This address 0004 where next address of PC 0033 is stored. Stack pointer shows it.

In practice, stack is operated by PUSH and POP instruction.

Let us take example of PUSH and POP instruction.

Before PUSH operation program counter is at 0056 and a JUMP instruction appears at this address, similarly stack contents are shown for previous subroutines 0051, 0080 etc. Stack pointer is stored with previous stack address 0007. Now after PUSH instruction, refer fig. (1.11) the contents of PC has incremented, by 1 to 0057. Then the address of stack pointer is also decremented by 2 it becomes 0005 from 0007 and now at this location of stack the contents of PC are stored at 0057 as shown. After completion of subroutine program counter should go to main program whose address is stored in stack by 0057. This is done by using instruction "POP".

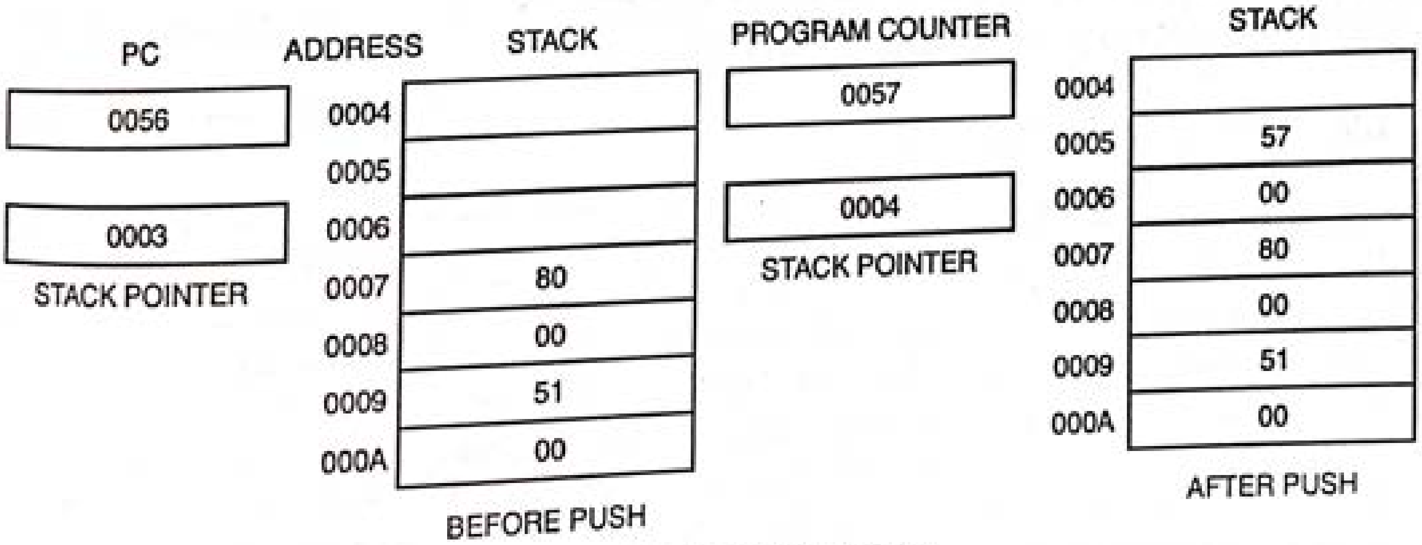


Fig. (1.11) Stack and Stack Pointer

Refer fig. (1.12), when "POP" instruction is given after completing subroutine it automatically stacks contents whose address was stored in stack pointer is 0005. It is transferred to program counter (0057). Similarly after "POP" operation stack pointer is incremented by 2 it becomes 0005 + 2 = 0007.

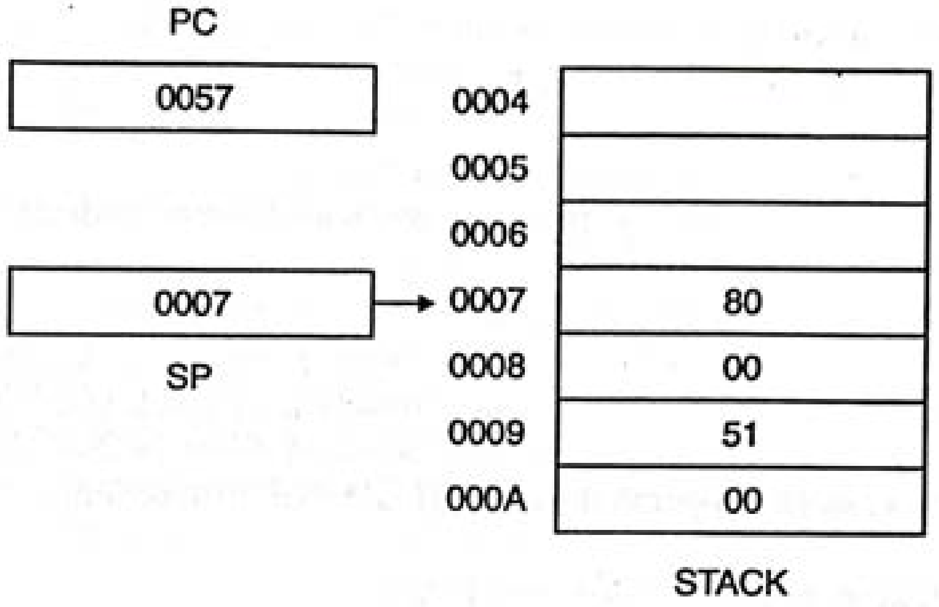


Fig. (1.12) Idea of POP Instruction

6. INCREMENTER – DECREMENTER

This is a 16-bit register it is used to add or subtract 1 from the contents of the program counter or stack pointer.

7. INSTRUCTION DECODER

Instruction is stored in instruction decoder. It is decoded by this circuit to initiate the timing and control signals for the given instructions. These operations are carried out by liming and control sections by generating control signals like CLKout, RD, $\overline{\mathrm{WR}}$ ALE S0, S1, IO/M, HLDA and RESET OUT. It also accepts input control signals like READY, HOLD and $\overline{\mathrm{RESET\ IN}}$.

1.9 INTERRUPTS

Microprocessor is always connected to different input/output (I/0) devices such as displays, printer, keyboard etc., which are called as peripheral devices. These devices communicate with microprocessor. If microprocessor asks to each device whether it requires service or not? Each time then the microprocessor will waste time that will reduce the speed of operation. Instead of that in microprocessor 8085 ***"interrupt"*** method is employed so that external device can interrupt the microprocessor whenever desired therefore there will be small wastage of time. When interrupt signal is given to the microprocessor it will suspend the normal sequence and it will give service to the device. After completing the service again microprocessor will continue its main program.

In microprocessor 8085 an interrupt is the input signal which stops the current execution and transfers to specific special routine known as *"Interrupt Service Routine"* (ISR). After ISR, microprocessor returns to the program. Fig. (1.13) illustrates the idea of communication between microprocessor and device with the help of interrupt.

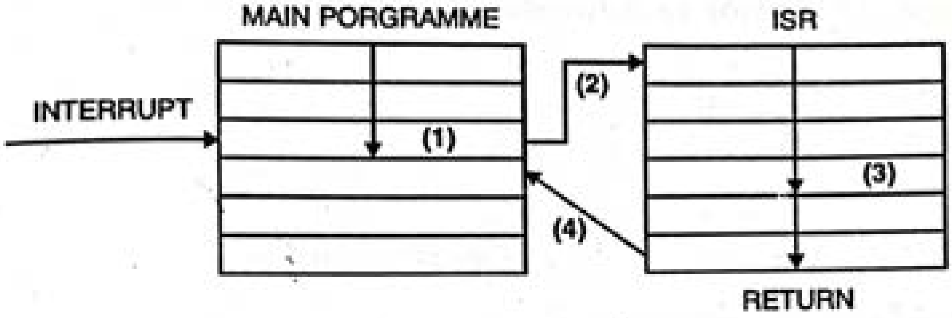


Fig. (1.13) Idea of Interrupt Service Routine

Suppose you are referring a telephone directory for finding the phone number of a person staring with surname X in sequence and if your friend told you that find a phone number of person starting with surname Y. Then your friend interrupts you; you will keep in mind the page number of your required person (like storing the address in Program counter). Then you will go to the page on which your friends phone number is given. You will find out the desired phone number for your friend, which is similar to *"interrupt service routine"*. After you service your friend again you will go for your page for finding the phone number starting with surname X, i.e. continuing main program.

Interrupts on 8085

The 8085 has two types of interrupts such as

(1) Software Interrupts, and
(2) Hardware Interrupts

It has eight software interrupts known as RSTO, RST1 ---- RST 7 where RST is RESTART because when this type of interrupt instruction is given to 8085 it jumps to specific fixed address. The address is calculated as suppose RST 1.

RST 1 = 1 x 8 =0008 H

When RST 1 interrupt is there 8085 goes to location whose address is 0008H for RST 5 it is 5 x 8 = 40 = 0028 H and so on. (40_{10} = 28H)

8085 microprocessor has five hardware interrupts as

1) TRAP (4.5) **Highest priority**
2) RST 7.5
3) RST 6.5
4) RST 5.5
5) INTR **Lowest priority**

These interrupts are vector interrupts it means when this type of interrupt is given to 8085 it is vectored or directed or transferred to specified fixed memory location. The address of these interrupts is calculated as

Interrupt	**Vector to Address**
TRAP	4.5 X 8 = 0024 H
RST 7.5	7.5 X 8 = 003C H
RST 6.5	6.5 X 8= 0034 H
RST 5.5	5.5 X 8 = 002CH

Where RST 5.5 = 5.5 X 8 = 44.0 = 44 in, decimal

(Where 44 = 2C in hexadecimal)

In these five interrupts TRAP has highest priority while INTR has lowest priority.

Similarly TRAP. is called as ***"nonmaskable interrupt"*** which can not be disabled microprocessor has to give service first for this interrupt. But other interrupts-RST 7.5, RST 6.5 RST 5.5 INTR are ***"maskable interrupts"*** these can be disabled.

Interrupt Procedure

(i) An I/O device sends an interrupt signal to the microprocessor to indicate that data is ready for input.

(ii) Microprocessor temporarily stops what it is doing and sends interrupt acknowledge signal.

(iii) Microprocessor stores the contents of program Counter into the stack and address of stack into stack pointer.

(iv) Microprocessor inputs the data from the device.

(v) It returns to the main program.

1.10 ADDRESSING MODES IN 8085

In the next chapter *'Instruction set of 8085'* you will study all types of assembly level instructions. An instruction is not recognised by microprocessor but it understands only binary code. Assembly level, instruction is indirectly converted into binary code that is known as *'machine code'*. Instruction may be 1, 2 or 3 byte in length. The first byte shows the operation type to be performed (operand) and second and third bytes of instruction indicates either the operand or address of the operand (data on which operation is to be performed).

There are number of instructions which are classified according to the type of addressing the data. They are classified into five types

(1) Direct Addressing

(2) Register Addressing

(3) Register Indirect Addressing

(4) Immediate Addressing

(5) Implicit Addressing.

Let us see each type of addressing mode with example.

1. DIRECT ADDRESSING

The direct addressing mode, the address of the operand is specified within the instruction itself. For example, LDA,06C2H. This means Load accumulator with the contents of memory location 06C2H. Here all instructions are three byte instructions. Another example of direct addressing is STA 08B2H

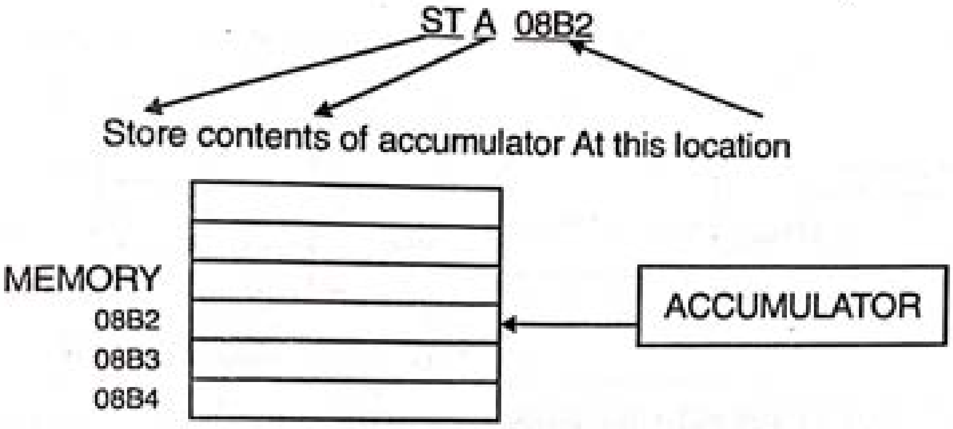

ST A 08B2
Store contents of accumulator At this location
MEMORY
08B2
08B3
08B4
ACCUMULATOR

Fig. (1.14) Direct Addressing

2. REGISTER ADDRESING MODE

In this mode the register name is specified in the instruction. Address is not required for this type. This is register to register or accumulator to register transfer.

These are single byte instructions. For example MOV A,B

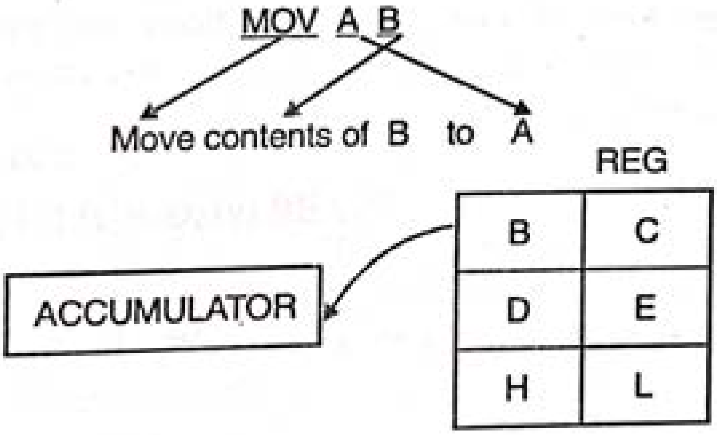

MOV A B
Move contents of B to A
REG
B
C
D
E
H
L
ACCUMULATOR

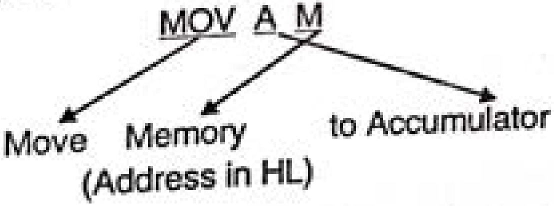
Fig. (1.15) Register Addressing

Another example is ADD C (Add contents of register C to accumulator). Here transfer is from Register C to Register A.

3. REGISTER INDIRECT ADDRESSING

Register indirect instructions are always one-byte instructions. Here register pair is specified for addressing 16 bit address of memory location.

For example, MOVA,M move memory contents to accumulator but memory location is specifically by H-L register pair


MOV A M
Move
Memory
(Address in HL)
to Accumulator

Another example of indirect addressing mode is ADD M

It adds the contents of memory whose address is stored in the HL pair to the accumulator. In actual program suppose we want to add the contents of memory location OFAFH to accumulator then instructions required are

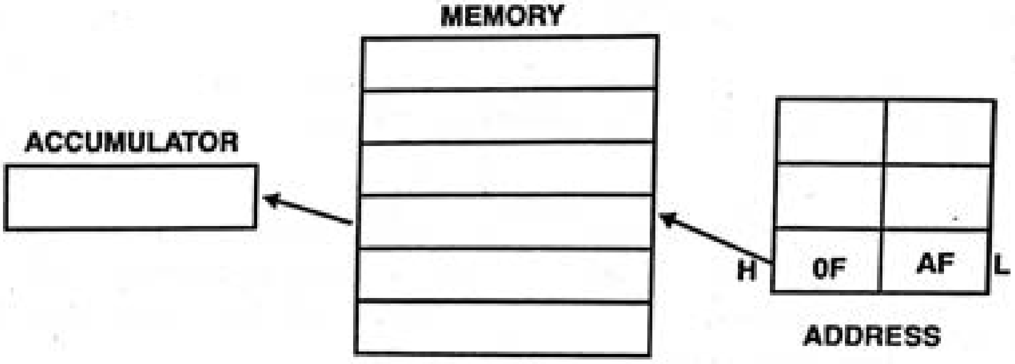


Fig. (1.17) Register Indirect Addressing

i) LXI H 0FAFH ; Load H-L pair with 0FAFH .
ii) ADD M ; Add contents of this memory (0FAFH) to accumulator.
(But MOV A, 0FAFH it is not allowed it is wrong instruction)

4. IMMEDIATE ADDRESSING

When data (operand) is directly transferred to register then it is known as immediate Addressing. One or two byte instructions are used for this type of addressing. In this mode operand is specified within the instruction itself. Data is specified after immediate Mnemonic.

Examples:

i) MVIB,55H (move immediate 55to B)
ii) LXI (Load register immediate) LXI H 52C5H

This .indicates load register H- L pair with data 52C5H, Here operand 52C5 is loaded in H-L registers.

iii)ACI 66H Add 66H to Accumulator with carry.

5. IMPLICIT ADDRESSING

Implicit addressing instructions are 1 byte instructions many instructions of logical group belong to this implicit addressing.

For example

i) CMA, Compliment Accumulator
ii) RRC Rotate Accumulator Right
iii) STC CMC etc.

The address of data is contained with the opcode itself and operand is absent.

QUESTIONS

1. **Select the correct alternatives**

a) Microprocessor 8000 is manufactured by ____ (Mar.95)

i) Intel ii)Motorola iii) Zilog iv) Toshiba

b) The example of a 32-bit Microprocessor is ____ (Mar.97)

i) M68000 ii)Z80000 iii) Intel's 80286 iv) M68020

c) The addressing mode of the Instruction LHLD is..... (Oct.98)

i) Register ii)Register Indirect iii) Direct iv) Immediate

d) LXI H, addr. is ________ byte instruction. (Specimen-Paper)

i) 1 ii)2 iii) 3 iv)4

e) The register in the CPU that keeps track of the address of the next instruction to be fetched from program memory is called the____ (Mar.2002)

i)Program Counter, ii) Instruction Register, iii) Accumulator, iv) Stack Pointer

f) In the flag register of 8085 microprocessor ____ number of bits are kept unused.

i)5, ii) 3, iii) 4, iv) 2 (Oct.03)

g) The instruction MOV B, A of 8085 microprocessor is an example of ______ addressing mode. (Oct.03)

i)Direct, ii) Implicit, iii) Register indirect, iv) Register

h) In 8085 microprocessor, serial data from external device is received on _____ pin.

i) SID, ii) SOD, iii) HOLD, iv) READY (Mar.2004)

i)— flag bit is reset, when flag resister content in D4H. (Mar.2006)

a) S b) Z c) CY d) AC

j)_____ bus is one way data path from MPU to all devices. (Mar.2008)

a) Data b) Address c) Control d) None of these

k)— is a non-maskable interrupt. (Mar.2010)

a) TRAP b) RST7.5 c) RST6.5 d) RST5.5

l) In 8085 —— pin is the only output terminal of interrupt control block (Mar.2012)

a) TRAP b) INTR c) RST7.5 d) $\overline{\text{INTA}}$

m) ALU is —— bit unit in 8085 microprocessor (Mar.2013)

i)8, ii)16, iii) 32, iv) 64

n) The flag register of 8085 contains —— flags. (Mar.2017)

a) 8 b) 3 c) 7 d) 5

2. Draw a neat labeled diagram of a generic micro-processor. (Oct.03,06,Mar.2008)

3. Draw a neat block diagram of the typical microcomputer system. Explain the function of each block in short. **(Mar 88,Mar.2019)**
4. Explain the primary functions of the CPU of a microcomputer. **(Mar.02, 04,06)**
5. What is Microprocessor ? Write the features/functions of 8085. **(Mar.2008,17,20)**
6. Draw internal block diagram of 8085. **(Mar.2010,2016)**
7. Write the functions of following Pins
   i)ALE ii)SID iii) CLK [out] **(Mar 88)**
8. What is the purpose of following registers in 8085 processors? **(Mar 88)**
   i) HL pair ii) Stack pointer iii) Program Counter
9. With the help of suitable examples, explain direct and immediate addressing modes in 8085. **(Mar 88)**
10. With suitable block diagram, explain the structure of CPU registers that are accessible to a user of 8085 processor. **(Mar 88)**
11. Explain the term vector interrupt. Why the vector interrupt pins on 8085 are named as RST 5.5, RST 6.5, RST 7.5? Which interrupt pin on 8085 has me highest priority? **(Mar 88,2016,17)**
12. Explain the-term 'interrupt'. What is the difference between interrupt pin INTR and other interrupts available on 8085 chip? Which interrupt of these can not be masked? **(Oct 88)**
13. List all hardware interrupts of 8085. List them according to their priority. Explain maskable and non-maskable interrupts. **(Mar.2006,07,16,19)**
14. Indicate the purpose of following pins on 8085 chip.
   i) ALE ii)SOD iii) $\overline{\text{WR}}$ **(Oct.88)**
15. What is the function of following register in 8085 processor?
   i) Accumulator . ii) Program counter iii) Flags iv) stack pointer **(Oct.88,Mar16)**
16. Explain the following blocks of 8085 microprocessor: **(Mar.2006)**
   i) Serial I/O Control ii) Accumulator iii) Multiplexed Address / Data Bus Buffer
17. Draw a block diagram of a microcomputer system. Explain in short the function of each block. Also show the direction of data flow among the various units of the microcomputer. **(Oct 88, Mar 89)**
18. Explain in brief how 8085 handles reading and writing in memory location and input/output ports. Discuss it with reference to the IO/$\overline{\text{M}}$ signal. **(Mar 89)**

19. Explain what do you mean by memory map of a system. Draw a memory map of a typical 8085 µp system. **(Mar 89)**

20. Explain the address and data bus structure of 8085. Show with help of a neat diagram how the address and data is demultiplexed? What is the advantage of such a bus ? **(Mar 89)**

21. Write a short note on "Evolution of Microprocessor" giving one example of each generation". **(Mar 2002,08, Oct.03,Mar.2020)**

22. How the data can be sent or received serially using the facility available on 8085 chip? Discuss the method. **(Oct 89)**

23. What is the function of following pins on 8085 chip?
    i) CLK ii)IO/M Hi) RESET OUT **(Oct 89, Mar.2004)**

24. Draw bit pattern of flag register and explain the significance of each flag bit. **(Mar.2006,2008)**

25. How many flag bits are present in the flag register? State the conditions when they are set to 1 in the execution of arithmetic instruction like ADP **(Oct 89)**

26. State any three addressing modes of 8085 with examples . **(Oct 89,Mar16)**

27. Draw a block diagram of 8085-microprocessor chip. **(Oct 89)**

28. Describe in brief functions of following pins in 8085. **(Mar 2002)**
    i)HOLD, ii) INTR, iii) $\overline{\text{RESET IN}}$

29. Explain any three flags available in 8085.If the flag register contains ACH interpret its meaning. **(Specimen Paper)**

27. Explain register indirect and immediate addressing modes in case of 8085 microprocessor with the help of suitable examples. **(Oct.03)**

28. Write the function of following units in the microprocessor 8085.
    i) Accumulator, ii) Stack pointer, **(Oct.03, Mar.2010,2019,2020)**
    iii) Instruction decoder, iv) Serial I /O control.

29. Describe in brief the function of following pins in 8085 microprocessor:
    i) HLDA, ii) READY, iii) RST7.5 iv) SID **(Oct.03,Mar16)**

30. Describe in brief the function of following pins in 8085 microprocessor:
    i) X_1 X_2, ii) A_{D0}-A_{D7}, iii) A_8-A_{15} **(Mar.2010)**

Answer Q.(1)
a) Zilog b)M68020 c)Direct d)3 e)Program counter f)3 g)register
h) SID i) CY j)Address k) TRAP l) INTA m) 8 n) 5 ❑❑❑

2

INSTRUCTION SET AND PROGRAMMING OF 8085

INTRODUCTION

As we all know, computers recognize and operate only with binary numbers. Each machine has its own instruction set based on the design of its CPU. Similarly for a microprocessor there exists its own instruction set. These instructions are in binary form and the language used is called machine language. Microprocessor design engineers select combinations of different bit patterns and assign a specific meaning to each combination by using electronic logic circuits. Each of these combinations is called as an instruction. Instructions are thus made up of one or more words. As we know a word is a group of bits. In the case of 8085 a word means an 8-bit group which is also called as a byte. A set of instructions designed into a machine makes up its machine language, which is specific to each machine. This chapter describes 8085 instruction set and its use in writing an assembly language programs.

2.1 INSTRUCTION CYCLE

Whenever any instruction is executed by a microprocessor number of different operations are to be carried out by the microprocessor. The clock achieves the overall control of the operations of microprocessor. Thus it may take several clock cycles to perform a particular operation. In the microprocessor terminology "the sub-division of an operation, which equals to one clock period is called as T-state". After defining the T-state we will define machine cycle and instruction cycle.

Machine Cycle

It is defined as "the time required to complete any operation, which is a sub-part of an instruction". The machine cycle may consist of number of T-states. For 8085 a machine cycle may consist of three to six T-states depending upon which machine cycle operation is being carried out. Following are the 8085-machine cycle operation,

No.	Machine Cycle Operation	Explanation of Operation
1.	Op-code Fetch	Fetches opcode from memory
2.	Memory read	reads data from memory
3.	Memory write	writes data into memory
4	Acknowledge	Acknowledges an interrupt.
7.	Halt .	Halts CPU while executing Halt instruction
8.	Hold .	Address data and control lines tri-stated for DMA operation.
9.	Reset	Reset operation after resetting.

Instruction Cycle

It is defined as "the time required to complete the execution of an instruction". An instruction cycle may consist of no. of machine cycles. For 8085 an instruction cycle may consist one to five machine cycles.

Thus any of the instruction when executed will require no. of machine cycles to be performed and which requires several T-states. A diagrammatic representation of these concepts can be shown as below

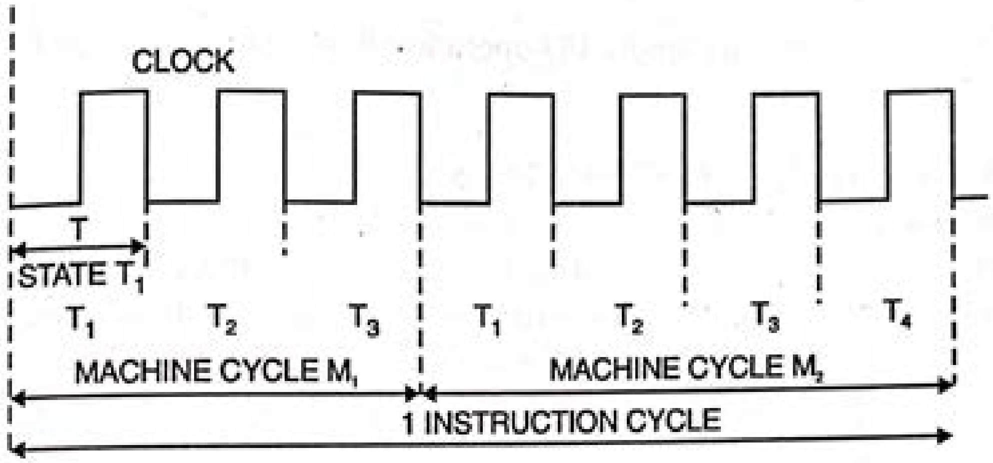


Fig. (2.1) Instruction Cycle

Consider fig. (2.1), which shows machine cycles T-states and Instruction cycle required for execution of a hypothetical instruction. From the diagram we see that the instruction requires two machine cyclesM_1 and M_2. Machine cycle M_1 lasts for three T-states while machine cycle M_2 lasts for four T-states.

2.2 INSTRUCTION CLASSIFICATION OF 8085

Instruction set of 8085 can be classified into the following five categories depending upon their function.

1) Data transfer group
2) Arithmetic operations group
3) Logical operations group
4) Branching operations group
5) Machine control operations.

1) DATA TRANSFER INSTRUCTIONS

This group of instructions copies data from a location called source to another location called destination, without modifying the contents of the source. In technical manuals the term data transfer is used for this copying function. However the term 'transfer' is misleading; it creates the impression that the contents of source are destroyed when, in fact, the contents are retained without any modification.

Some of the various types of data transfer operations available in 8085 are as follows

	Types of Transfer	Example
1)	Between registers	Copy contents of register B to register E
2)	Specific data byte to register	Load register C with data byte or memory location 4AH
3)	Between memory location and a register	Copy data from memory Location 1500H to register D
4)	Between I/O device and accumulator (I/O operations)	Input from a keyboard to accumulator

The last type of operation namely I/O operation is sometimes grouped in separate group called I/O instructions.

2) ARITHMETIC GROUP INSTRUCTIONS

These instructions perform arithmetic operations such as addition, subtraction, increment, and decrement. 8-bit as well as 16-bit addition can be performed in 8085. 8-bit 2's complement subtraction is performed in 8085. Also 8-bit contents of. register /memory location as well as 16-bit contents of register pair or stack pointer can be incremented or decremented using instructions in this group.

3) LOGICAL GROUP INSTRUCTIONS

These instructions perform following logical operations. In all the operations accumulator is one operand and where required second operand can be register content, memory content or 8 - bit data.
The results are stored in accumulator

a) AND operation
b) OR operation
c) XOR operation
d) Rotate
e) Compare
f) Complement or NOT operation.

4) BRANCHING GROUP INSTRUCTIONS

These instructions allow user to alter the sequence of execution of the program either conditionally or unconditionally.

Following are the types of Branch instructions available in 8085.
(a) Unconditional Jump
(b) Conditional Jump
In this case a jump is taken after checking for given condition.

(c) Call and Return instructions - These are the instructions used to enter into and return from a subroutine.
(d) Restart instructions - These are used to enter respective Interrupt service routine.

5) MACHINE CONTROL INSTRUCTIONS

These instructions are for following operations

a) Halt
b) Set interrupt mask
c) Read interrupt mask
d) No operation - (Do nothing)
e) Stack operations

2.3 INSTRUCTION FORMAT

As explained earlier an instruction is a combination of bit pattern. Normally each instruction can be viewed as a collection of two parts. One part, which gives the task to be performed, is rightly called as OPERATION CODE (Opcode) and the other part gives the data to be operated on called as OPERAND. The operand can be specified in various ways.

Instructions for 8085 are of three formats.

i) One byte instructions ii) Two byte instructions iii) Three byte instructions

I) One Byte Instructions

These are the instructions, which include opcode and operand both in the same byte. In same cases the operand may be implicit. Such a type of instruction requires a single memory location. We will see examples of these types of instructions in next sections while learning the instruction set in detail. e.g. RRC, MOVB,C

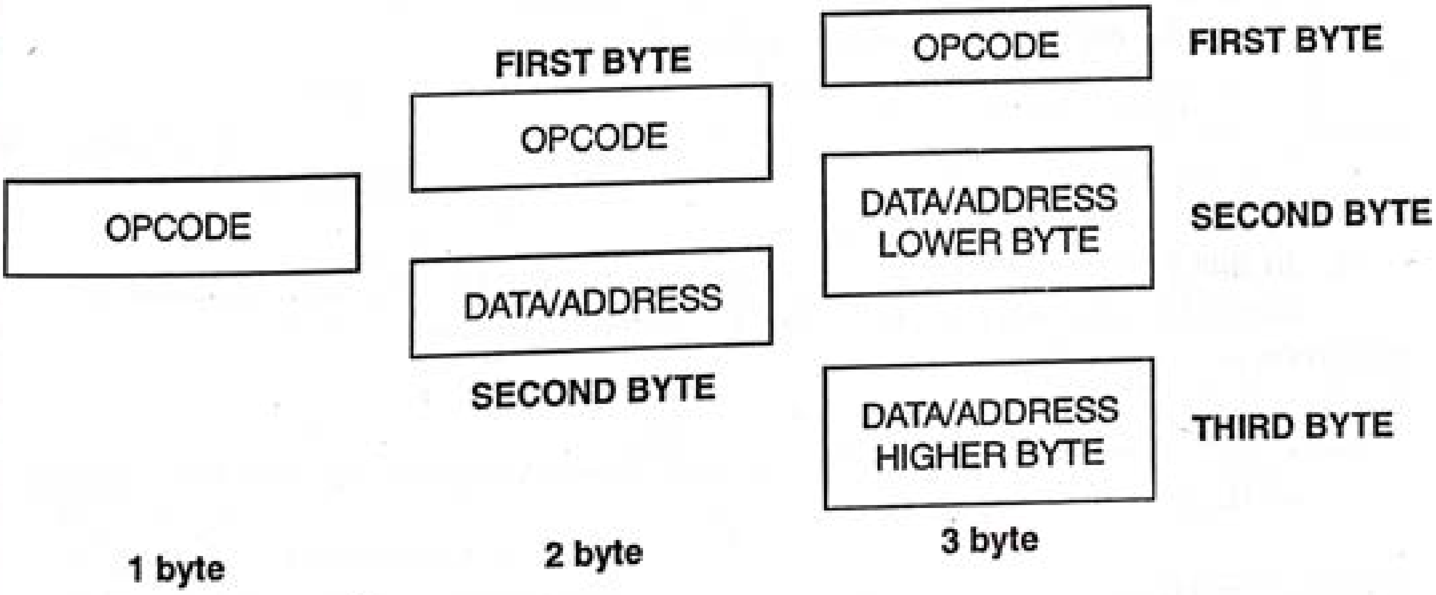

FIRST BYTE
OPCODE
FIRST BYTE
OPCODE
OPCODE
DATA/ADDRESS LOWER BYTE
SECOND BYTE
DATA/ADDRESS
SECOND BYTE
DATA/ADDRESS HIGHER BYTE
THIRD BYTE
1 byte
2 byte
3 byte

Fig. (2.2) Instructions Formats

II) Two Byte Instructions

In a two-byte instruction, the first byte specifies opcode and the second byte specifies the operand. Such a type of instruction would require two memory locations to store in memory. Next few sections will explain about such instructions.

e.g. MVI A,01H MVI A,(opcode-first byte) 01H(second byte)

III) Three Byte Instructions

In this type of instruction the first byte specifies the opcode as usual but the second and the third bytes together specify 16-bit address of operand or 16-bit data. But the important fact is the second byte is lower order byte and the third byte is higher order byte. Such an instruction would require three memory locations to store in memory. The examples and explanation for these type of instructions follows in next few sections. e.g. LXIH,2005H

LXIH, ,(opcode-first byte[21]) 05H(second byte) 20H(third byte)

2.4 DATA TRANSFER INSTRUCTIONS

As seen earlier these instructions allow user to copy data from source to destination. Following are the instructions in this group.

1) MOVE INSTRUCTION

This instruction copies contents of source register to destination register. If one of the operands is a memory location then its address is specified in **HL register pair.** No flags are affected. The formats of these instructions are as follows.

Opcode	Operand	Bytes	Machine Cycles	T-states
MOV	Rd, Rs	1	1	4
MOV	M, Rs	1	2	7
MOV	Rd, M	1	2	7

Rs = Source location register. Rd = Destination location register.
Rd and Rs can be any of the following registers A, B, C, D, E, H, L.
M = Memory location whose address is given by contents of HL pair

Important Note:

1. The data in assembly language is always followed by ‘H’ e.g. 3AH ‘H’ stands for Hexadecimal number.
2. In this topic each instruction is specified with bytes, T states machine cycles this is for only information. It need not be remembered.

Examples:

MOV A, B ; Copy contents of B into A

MOV A, M ; Copy contents of memory location whose address is given by contents of HL register pair into accumulator.

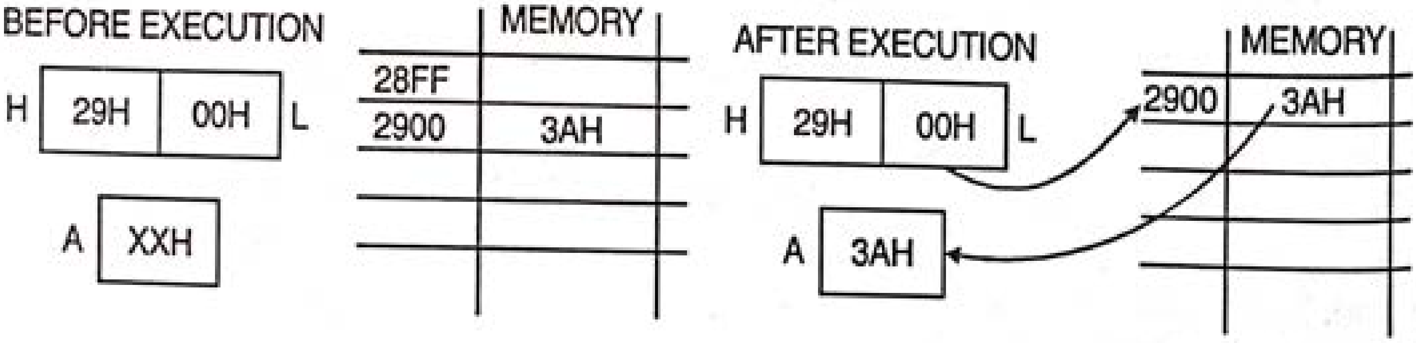


Fig. (2.3) Idea of MOV A,M

2) MVI (MOVE IMMEDIATE) INSTRUCTION

This 2-byte instruction is used to store the 8-bit data, given in instruction itself, into the destination, which may be a register or memory location. The first byte of instruction gives the destination and second byte is the 8-bit data. No flags are affected.

Formats for these instructions are as follows.

Opcode	Operand	Bytes	Machine Cycles	T-states
MVI Rd,	data	2	2	7
MVI M,	data	2	3	10

Rd : destination register. Which can be any one of the following A, B, C, D, E, H, L
Data : 8-bit immediate data
M : memory location pointed by HL contents.

Examples :
MVI B, 3CH ; Store 3CH into register B.
MVI M, 50H ; Store 50H into memory location pointed by HL Contents.
MVI D, 5FH ; Store 5FH into register D.

3) LDAX – LOAD ACCUMULATOR INDIRECT

This is a single byte instruction. One of the operands is implicit and is accumulator. The other operand is a register pair. It is designated in instruction itself. The contents of the designated register pair point to a memory location. This instruction copies the contents of that memory location into the accumulator. The contents of either the register pair or the memory location are not altered. No flags are affected.

Opcode	Operand	Bytes	Machine Cycles	T-states
LDAX	rp	1	2	7

rp : register pair.
rp : B means BC register pair is used as pointer.
rp : D means DE register pair is used as pointer.

Example:
If BC register contents are 2500H i.e. B = 25H and C = 00H
and if LDAX B instruction is executed
then accumulator will be loaded with the contents of the memory location 2500 H.

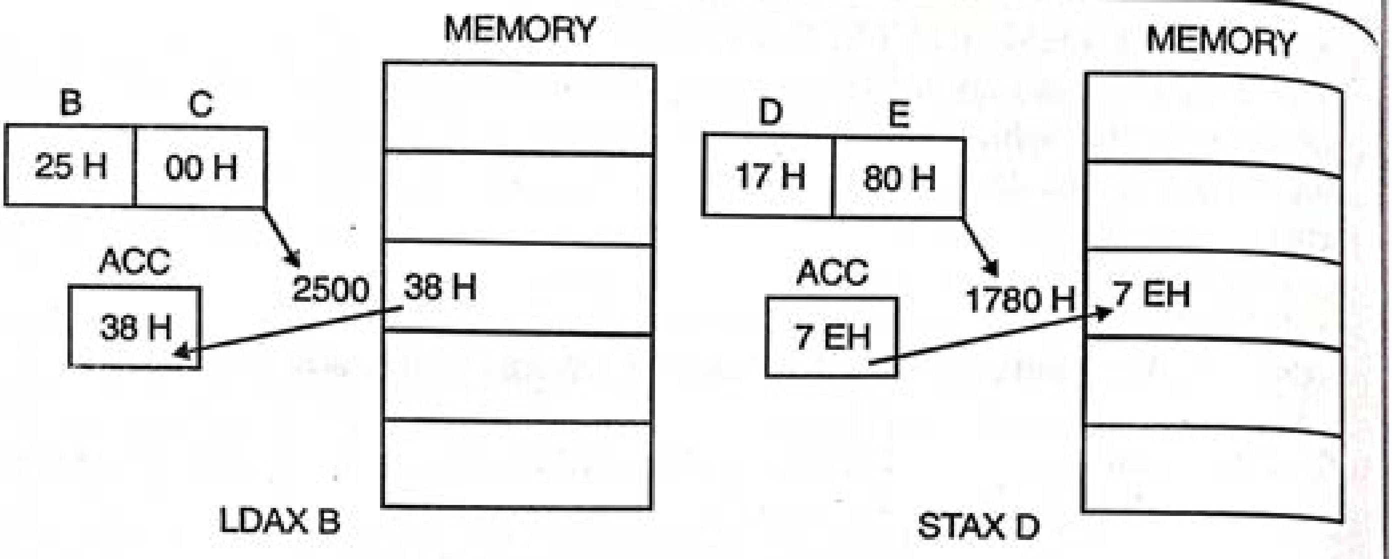


Fig. (2.4) Illustration for LDAX & STAX instructions

4) STAX – STORE ACCUMULATOR INDIRECT

This is single byte instruction one of the operands is implicit and it is accumulator. The other operand is register pair. It is designated in instruction itself. The contents of the designated register pair point to a memory location. This instruction copies the contents of the accumulator into the memory location pointed by register pair. The contents of either register pair of the accumulator are not altered. No flags are affected.

Opcode	Operand	Bytes	Machine Cycles	T-states
STAX	rp	1	2	7

rp : register pair.
rp = B means BC register pair acts as pointer.
rp = D means DE register pair acts as pointer.

Examples :

If contents of Accumulator are 7EH and D register contents 17H and E register contents 80H then after execution of the instruction STAX D the contents of memory location 1780 H will be 7EH.

5) LDA – LOAD ACCUMULATOR DIRECT

This is a 3 byte instruction, first byte gives opcode and the second and third provide a 16-bit address for a memory location. Second byte specifies lower order byte of provide a 16-bit address for a memory location. Second byte specifies lower order byte of address and third byte specifies the higher order byte of address. This instruction when executed copies contents of the memory location pointed by address given in instruction to the accumulator. Contents of source are not altered. No flags are affected.

Opcode	Operand	Bytes	Machine Cycles	T-states
LDA	addr	3	4	13

addr : 16-bit address.

Example :

If memory location 3780 H contains FEH and the instruction

LDA 3780H

is executed, then after the execution contents of accumulator will be FEH. The contents of memory location 3780 H are not altered. While writing this instruction in Hex code, we will write in following order

(opcode), 80H, 37H

OR (3A) , 80H, 37H where 3A is opcode of LDA

Where (opcode) is 8-bit machine code for the instruction?

6) STA –STORE ACCUMULATOR DIRECT

This is also a three-byte instruction. The first byte gives opcode and the second and third byte provides a 16-bit address which points to a memory location. As in the case of LDA instruction the second byte specifies lower order byte of the address and the third byte specifies higher order byte of the address. This instruction when executed copies the contents of the accumulator into the pointed memory location. Accumulator contents are not destroyed. No flags are affected.

Opcode	Operand	Bytes	Machine Cycles	T-states
STA	addr	3	4	13

addr : 16-bit address

Example :

If accumulator contents are 19H then executing the instruction

STA 2727H

Will copy 19H into the memory location 2727H. The contents of accumulator are not altered.

7) LXI – LOAD REGISTER PAIR IMMEDIATE

This is a 3-byte instruction first byte designates op-code and the register pair operand. The second and third byte specify a 16-bit data with usual convention of lower order byte first and higher order byte next. On execution this 16-bit data is copied into the designated register pair. No flags are affected.

Opcode	Operand	Bytes	Machine Cycles	T-states
LXIrp,	data	3	3	10

rp : register pair

rp can be any one register pair of the following.

Indication	Implied reg-pair
B	BC
D	DE
H	HL
SP	SP

Data : 16-bit data

Example:

LXI H, 2100 H will be written in the order (opcode), OOH, 21H. Upon execution H register will contain 21H and L register will contain will OOH.

(Instruction LXI in this case 'X' indicates there is a pair. Remember; if 'X' is present in any instruction that means there is a pair.)

8) LHLD – LOAD H AND L REGISTER DIRECT

This is also a 3-byte instruction. The first byte gives opcode. The second and third bytes specify a 16-bit address in usual convention of lower order byte first and higher order byte next. On execution the contents of memory location pointed by the specified address are copied into L register. The contents of the memory location next to specified location is copied into H register. Contents of these memory locations are not altered. No flags are affected.

Opcode	Operand	Bytes	Machine Cycles	T-states
LHLD	addr	3	5	16

addr : 16-bit address.

Example :

Assume memory location 2057H contains 55H and 2058H contains 25H then to transfer these to HL pair we will execute the instruction.

LHLD 2057H

Then after execution of above instruction

Register H will contain 25H and Register L will contain 55H

9) SHLD – STORE H AND L REGISTER DIRECT

This also is a 3-byte instruction. The first byte gives opcode. The second and third byte specify a 16-bit address in usual convention of lower byte first and higher order byte next. On execution the contents of L register are copied to the memory location pointed by the address specified in instruction and the contents of H register are copied to the memory location next to the specified location. Contents H and L registers are not altered. No flags are affected.

Opcode	Operand	Bytes	Machine Cycles	T-states
SHLD	addr	3	5	16

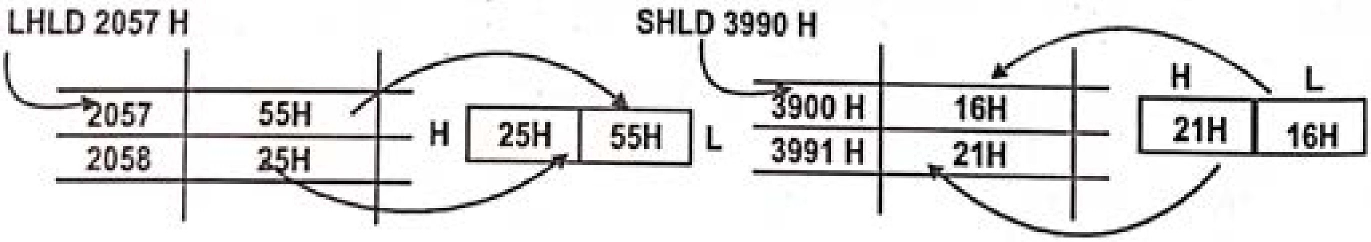


Fig. (2.5) SHLD & LHLD Instructions

addr : 16-bit address.

Example :

Assume contents of HL register pair before execution as H = 21H, L = 16H

After execution of the instruction SHLD 3990H

We will have contents of memory location as refer fig.(2.5)

Memory location	Data
3990H	16H
3991H	21H

10) XCHG – EXCHANGE H AND L WITH D AND E

This is a single byte instruction. On execution of this instruction contents of H register are exchanged with contents of D register and contents of L register are exchanged with contents of E register. No flags are affected.

For this instruction the operands are implicit and no explicit operand is written.

Opcode	Operand	Bytes	Machine Cycles	T-states
XCHG	none	1	1	4

Example: Suppose contents of HL register pair and DE register are as follows.

Contents before execution of instruction:

Register	Data	Register	Data
H	15H	L	70H
D	25H	E	90H

But after executing the XCHG instruction the contents of these registers will be as follows. Contents of register after execution of XCHG instruction:

Register	Data	Register	Data
H	25H	L	90H
D	15H	E	70H

11) IN-INPUT 8-BIT DATA FROM AN INPUT PORT TO ACCUMULATOR.

This is the instruction which is included in the I/O instruction group. This is a two-byte instruction. First byte signifies the operation code for "input data from Port" operation. The second byte gives the 8-bit port address. When this instruction is executed, the microprocessor sends the 8-bit port address on lower address bus A0-A7. It also duplicates this address on higher address bus A8-A15. Any one of the set i.e. lower or higher address bus, can be decoded to enable the input port. The 8-bit data is then inputted from selected port into accumulator. No flags are affected.

In 8085 ports address can range from OOH to FFH. Which allows the user to have at the most 256 ports configured.

Opcode	Operand	Bytes	Machine Cycles	T-states
IN	Inport	2	3	10

Inport: 8-bit port address

Example: Consider the instruction IN 5BH

When this instruction is executed the microprocessor reads 8-bit data present at the input port 5BH.

12) OUT – OUTPUT 8-BIT DATA FROM ACCUMULATOR TO AN OUTPUT PORT

This is the other instruction, which is included in the I/O instructions group. This also is a two-byte instruction. The execution of this instruction is similar to the execution of IN instruction except that the direction of data transfer in this case is from microprocessor to output port. In other words the 8-bit data in accumulator is output on to the output port. As in the case of IN instruction the first byte signifies opcode and the second one informs the 8-bit output port address. No flags are affected.

Opcode	Operand	Bytes	Machine Cycles	T-states
OUT	Out port	2	3	10

Output port: 8-bit output port address.

Example: Consider the instruction OUT 27H

When this instruction is executed the contents of accumulator are output to the output port 27H.

This completes the discussion of Data transfer group of instructions. In the next two sections we will try to write simple 8085 assembly language programs. We will see how to write a correct and systematically written language program.

2.5 HOW TO WRITE AND EXECUTE A SIMPLE ASSEMBLY LANGUAGE PROGRAM

While working on a microprocessor we are required to write different programs very frequently. As you know now the microprocessor only understand, binary information in coded form, called machine code. Thus to think of writing a program we should have it written in machine codes. But writing direct machine code programs is not only tedious but also makes one uneasy and reduces program development speed. As a solution to this problem the instructions in programs are written not in direct machine code but are in mnemonic form. This is called assembly language. Once written in assembly language the program can be coded in machine codes (Machine language).

To think of writing assembly language programs requires that the programmer follow the path of logical thoughts. A given task should be broken into subtasks and programming to carry out there subtasks can then be done independently. This is called **modular design** approach.

Writing a program is equivalent to giving specific commands to the microprocessor in sequence, to perform a task. this process can be divided into following steps.

Step 1: Read the problem carefully
Step 2: Break it down into small steps
Step 3: Represent these small steps in possible sequence with a flowchart
Step 4: Translate each block in flowchart into appropriate mnemonic (assembly language) instructions

Step 5: Translate assembly language program to machine code (machine language).

1 to 4 accounts for assembly language program development.

Step 5 along with steps 1 to 4 accounts for the development of executable machine language program.

Illustration: We will now apply this approach to solve following problem.

Problem statement: Write a program in 8085 assembly language to load the contents of D register in C register and display the same at an output port FEH.

Problem analysis: The problem can be broken into following subtasks.

Step no.	Task
1	Load register C with contents of D
2	Output contents of D at O/P port FEH

We now ask following questions for the step indicated.

a) For Step 1: Is there an instruction available to load register C with contents of register D

Answer: Yes, Use MOV C, D

b) For Step 2: Is there an instruction which will output contents of register D at output port FEH

Answer: No, so subdivide Task 2 as

Task 2a Load contents of D in A

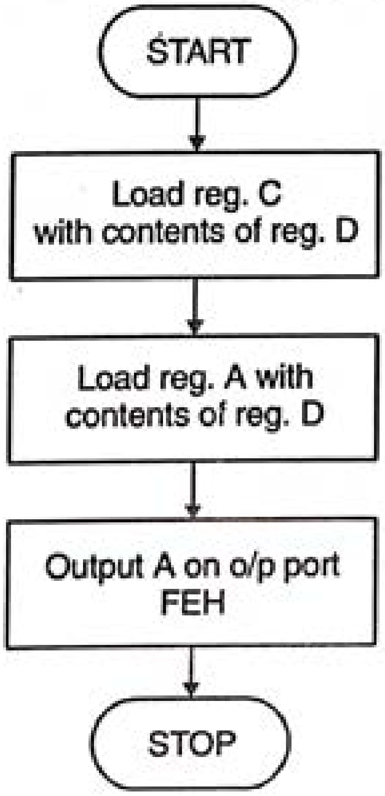
Task 2b Output A to O/P port FEH

For Task 2a use MOV A, D

For Task 2b use OUT FEH

Flowchart:

We now draw the flowchart symbolizing flow of actions to be taken. Following are the symbols used for flowcharts with their appropriate meanings.


START
Load reg. C with contents of reg. D
Load reg. A with contents of reg. D
Output A on o/p port FEH
STOP

Program:

The final assembly language program can be written in a standard format this format includes following columns.

a) Address
b) Machine code
c) Label
d) Mnemonic
e) Operand
f) Comments

Normally columns c,d,e,f are prepared first and after coding the program columns a and be are completed. We will only write columns c,d and e,f.

Label	Mnemonic	Operand	Comments
Start:	MOV	C,D	; Load reg. C with reg. D contents
	MOV	A,D	; Load accumulator with reg. D contents
	OUT	FEH	; Ouput Acc on FEH
	HLT	-	; Stop processing

Illustration: Program for data transfer from memory to memory.

1. Write a program to transfer one byte of data from memory location 2005 H to 4020 H

Solution:

Program:

Label	Mnemonic	Operand	Comments
START	LXI H,	2005 H	; Load HL pair with 2005 H
	MOV	A,M	; Transfer memory contents to Acc
	LXI H,	4020 H	; Load HL pair with 4020 H
	MOV	M	; Transfer Acc to memory
	HLT	-	; Stop processing

Important Note: Students should note that direct memory to memory data transfer instruction is not available in 8085.

Exercises

i) Write a program in 8085 assembly language to store EFH in following registers Registers: B,C,D,H,L
(Hint: use MOV and MVI instructions)

ii) Write a program in 8085 assembly language to copy contents of memory location 3700 H to E register as well as to memory location 4700H
(Hint: Use LXI H, MOV instructions)

iii) Write a program in 8085 assembly language which will input one byte of data from input port 37H and store it at a memory location 3500H
(Hint: Use IN, STA instructions)

iv) Write a program to output one byte stored at memory location 47FEH to the output port 3FH.
(Hint: Use OUT, LDA instructions)

v) Complete the following program by adding appropriate comments to each instruction.

Program	Comment
LXI D 3330 H	-
LXI H 2300 H	-
MOV A, M	-
OUT 39 H	-
IN 49 H	-
STAX D	-
HLT	-

2.6 ARITHMATIC GROUP INSTRUCTIONS

As discussed earlier this group contains instructions, which allow arithmetic operations such as addition, subtraction, increment and decrement along with certain special instructions. Following are the instructions in this group.

1) ADD – ADD REGISTER TO ACCUMULATOR

This is a single byte instruction with operand placed along with opcode in the code byte. The operand can be a register or a memory location pointed HL register pair.

When this instruction is executed the contents of operand i.e. either register or memory location contents are added to the accumulator or the result is stored in accumulator. All flags are modified to reflect the result of addition. This addition is an 8-bit unsigned binary addition. The instruction has following formats.

Opcode	Operand	Bytes	Machine cycles	T-states
ADD	reg	1	1	4
ADD	M	1	2	7

reg: any one of the following registers
M: memory location pointed by contents of the HL register pair.

Examples:

a) Consider the instruction ADD D

Let the contents of registers A and D before execution of above instruction be

D = 51H A = 47H

On execution of the ADD D instruction following addition is carried out

$$\begin{array}{rcl} 47\text{ H} & = & 0100\ \ 0111 \\ +51\text{ H} & = & 0101\ \ 0001 \\ \hline 98\text{ H} & = & 1001\ \ 1000 \end{array}$$

then sign flag = S = 1, (Since D7 bit of result is 1)
Zero flag = Z = 0 (Refer table of flags)
Aux Carry = AC = 0, Parity Flag = P = 0 Carry flag = C = 0
Thus contents of A,D and flags register, after the execution are as follows.

	S	Z	X AC	X P	X C
A = 98 H	1	0	0 0	0 0	0 0
D = 51 H			Flags =	80 H	

b) Consider the instruction ADD M

Let contents of A,H,L registers before the execution of above instruction be

A = 76 H , H = 25 H , L = 35 H

Let contents of memory location 2535 H be A2 H then after execution of ADD M A2 H will be added to 76 H and result will be stored in Accumulator. All flags will be affected according to result the contents will be

A 18 H
H 25 H
L 35 H

S	Z	X	AC	X	P	X	C
0	0	0	0	0	1	0	1

Flag = 05H

2) ADI – ADD IMMEDIATE TO ACCUMULATOR

This is a two by instruction. First byte gives opcode. The second byte is itself 8-bit data. This immediate data is added to accumulator contents. Result is stored in the accumulator and all flags are modified. This type of addressing mode is termed as an immediate addressing.

Opcode	Operand	Bytes	Machine cycles	T-states
ADI	data	2	2	7

data: 8-bit immediate data.

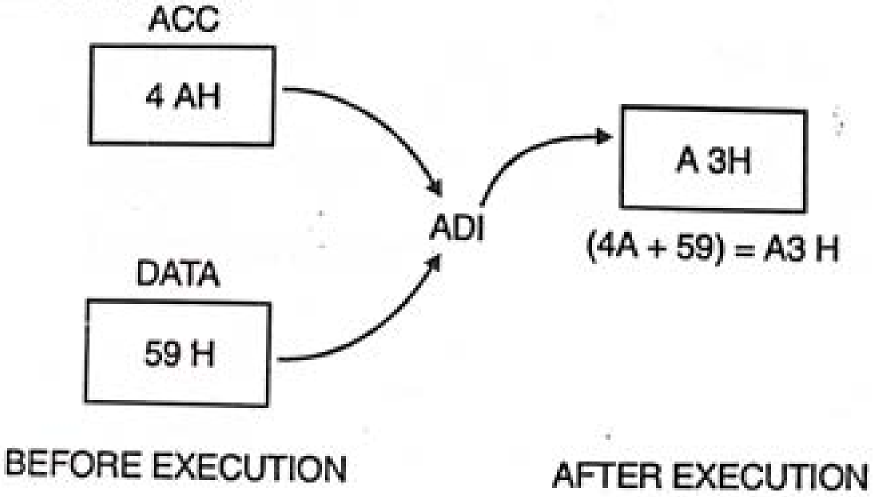


Fig. (2.6)

Example:

Let contents of A be 4A H . Consider the instruction ADI 59 H

After execution of above instruction the contents of accumulator will be A=A3 H and flags will be 95 H (after converting flag contents in to Hex).

S	Z	X	AC	X	P	X	C
1	0	0	1	0	1	0	0

3) ADC – ADD REGISTER TO ACCUMULATOR WITH CARRY

This also is a one-byte instruction. The operand, which is placed along with opcode in the code byte, can be either a register or the memory location pointed by HL pair. On execution of this instruction, contents of operand and carry flag are added to accumulator and the result is stored in Accumulator. All flags are modified. This instruction is generally used for 16-bit addition.

Opcode	Operand	Bytes	Machine cycles	T-states
ADC	reg	1	1	4
ADC	M	1	2	7

reg: any of the following registers B, C, D, E, H, L, A

M: memory location pointed by HL reg. pair

Example: Let contents of Accumulator, Carry flag, B register, before the execution of the instruction ADC B be

A = 98 H B = A1H Carry flag = 1

When ADC B is executed we get the result stored in accumulator. A = 3A H & carry flag = 1

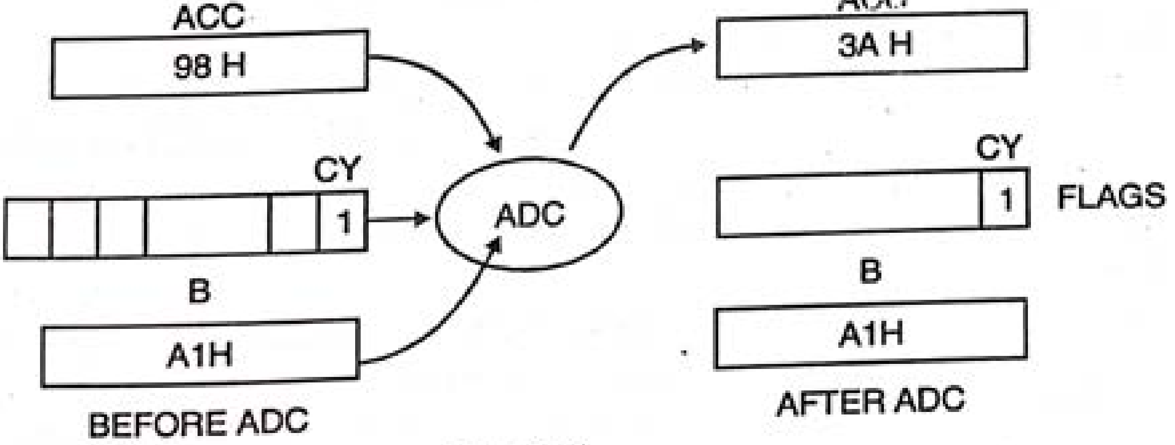


Fig. (2.7)

4) ACI-ADD IMMEDIATE TO ACCUMULATOR WITH CARRY

This also is a two-byte immediate addressing mode instruction when it is executed, the 8-bit data (Second byte of instruction) is added along with carry to accumulator contents and the result is stored in accumulator. All flags are modified according to result.

Opcode	Operand	Bytes	Machine cycles	T-states
ACI	data	2	2	7

data: 8-bit immediate data.

Example:

Let following be some of the register contents.

A=26 H Carry = 1

then after execution of the instruction

ACI 57 H

the contents will A = 7EH Carry flag = 0

5) SUB-SUBTRACT REGISTER OR MEMORY FROM ACCUMULATOR

This is also a single byte instruction when executed, works almost same as ADD instruction except that in this case memory or register contents are subtracted from accumulator and the contents of source are unaltered. All flags are modified to reflect result of subtraction. **CY flag is complimented after subtraction.**

Opcode	Operand	Bytes	Machine cycles	T-states
SUB	reg	1	1	4
SUB	M	1	2	7

reg: any of the following registers B, C, D, E, H, L, A

M: Memory location pointed by HL pair

Example:

Consider the contents of Accumulator and register E before execution of the instruction SUBE

Contents before execution

A = 40 H = 0100 0000 and E = 37 H = 0011 0111

1's complement = 1100 1000

2's complement = 1100 1000 +1 =1100 1001

A+ E' = 0100 0000 + 1100 1001 =**1** 0000 1001 =09H

CY =1 after complementing CY =0

then when the instruction SUB E is executed a 2's complement subtraction is carried out as shown.

If result is positive the carry flag is **reset** and the accumulator contains the result. If results negative the carry flag is set and the accumulator contains 2's compliment of the magnitude of result. In this case contents of Accumulator and carry flag will be

A= 09 H Carry = 0

6) SUI-SUBTRACT IMMEDIATE FROM ACCUMULTOR

This instruction is also a two-byte instruction. It uses the immediate addressing mode. The first byte gives opcode and second byte gives immediate data. The immediate data is subtracted from the contents of accumulator and the result is stored in accumulator. All flags are modified according to the result. The subtraction is a 2's complement subtraction and the significant of result is same as that for the SUB instruction.

Opcode	Operand	Bytes	Machine cycles	T-states
SUI	data	2	2	7

data: 8-bit immediate data

Example:

Let the accumulator contents be 40 H. If we execute this instruction

SUI 37 H

then accumulator will contain 09 H with carry flag = 0 indicating that the result is positive. (*Perform this subtraction by 2's complement method*)

7) SBB-SUBTRACT SOURCE AND BORROW FROM ACCUMULATOR

This also is a single byte instruction. It subtracts contents of source and borrow from the accumulator contents. It is a 2's complement subtraction. Result is stored in accumulator. All flags are modified according to the result. The source can be either any of the registers or any memory location. The memory location to be used, as source should be pointed by HL register pair. The carry flag acts as borrow.

Opcode	Operand	Bytes	Machine cycles	T-states
SBB	reg	1	1	4
SBB	M	1	2	7

reg: any of the following registers B, C, D, E, H, L, A

M: memory location pointed by HL pair

Example:

Let contents of Accumulator, D register and carry flag be

A = 37 H, D = 3F H and Carry = 1

After execution of the instruction SBB D

We get result in accumulator as A = F7 H with carry flag = 1

Carry flag set indicates that result in accumulator is negative and is in 2's complement form.

8) SBI-SUBTRACT IMMEDIATE WITH BORROW

This is a two-byte instruction using immediate addressing mode. The first byte is opcode and second byte is 8-bit immediate data. The immediate data and borrow is subtracted from accumulator contents. It is a 2's complement subtraction. The result is stored in accumulator. The significance of result is same as that for SUI instruction. All flags are modified.

Opcode	Operand	Bytes	Machine cycles	T-states
SBI	data	2	2	7

data: 8-bit immediate data

Example:

Consider the instruction SBI 20 H with initially set carry flag acting as borrow When executed it will subtract 21 H from accumulator contents and store the result in accumulator.

9) INR-INCREMENT CONTENTS OF REGISTER/MEMORY BY 1

This is a single byte instruction when executed the contents of operand are incremented by 1. **All flags except carry flags are modified.** The operand can be any of the register or a memory location pointed by HL register pair.

Opcode	Operand	Bytes	Machine cycles	T-states
INR	reg	1	1	4
INR	M	1	3	10

reg: any of the following registers B, C, D, E, H, L, A
M: memory location pointed by HL pair

Example:

Let contents of E register be E = 2F H 0010 1111
If we execute the instruction INR E + 1
0011 0000 = 30H

then the modified E register and flags will be as follows

E = 30 H = 0 011 0 0 0 0 Flag = 14H

Flags									
	0	0	0	1	0	1	0	0 N/C	C=NC=last state
	S	Z	X	AC	X	P	X	C	

10) DCR-DECREMENT SOURCE BY 1

This is also a single byte instruction, when executed will decrement the contents of source by1 .All flags except carry are modified. The source can be any register or memory pointed by HL pair.

Opcode	Operand	Bytes	Machine cycles	T-states
DCR	reg	1	1	4
DCR	M	1	3	10

reg: any of the following registers B, C,D, E, H, L, A

Example:

Contents of Accumulator
before execution A 55 H

Contents of Accumulator
and flag after execution A 54 H

Flags								
	0	0	X	0	X	0	X	NC
	S	Z		AC		P		C

11) INX-INCREMENT REGISTER PAIR BY 1

This is a single byte instruction. Which when executed increments the 16-bit contents of operand register pair by 1. **No flags are affected.** This is also called as extended increment.

Opcode	Operand	Bytes	Machine cycles	T-states
INX	rp	1	1	6

rp : any of the following 16-bit operand (reg. pairs)

e.g if HL = 29FFH
then after INXH HL = 3000H (29FF+1 = 3000H)

(Remember: The difference between INR and INX: INR is used for register and INX is used for register pair. X represents Pair)

12) DCX-DECREMENT REGISTER PAIR BY 1

This also is a single byte extended decrement register. It decrements the contents of operand (16-bit) quantity by 1. **No flags are affected.**

Opcode	Operand	Bytes	Machine cycles	T-states
DCX	rp	1	1	6

rp: same significance as rp in INX instruction

e.g if BC = 40 01H
then after DCXB BC = 4000H (4001-1 = 4000H)

13) DAD-ADD REGISTER PAIR TO H AND L REGISTERS

This is also a single byte instruction. It adds 16-bit contents of specified operand to 16-bit contents of H and L register pair. The result is stored in H and L register pair. **Only carry flag is affected.** No other flags are affected. Carry flag sets when result is greater than 16-bit number. This instruction is also called as double add instruction. Contents of operand are unaffected.

Opcode	Operand	Bytes	Machine cycles	T-states
DAD	rp	1	3	10

rp: reg. pair operands same as rp for INX instruction

Example:

Assume contents of HL pair and SP as follows

H = 10 H L =10 H
S = 23H P = 23 H

If we execute the instruction DAD SP then modified contents of H, L and SP are as follows

H =33 H L = 33 H carry flag is reset
S = 23H P =23 H no other flags are affected

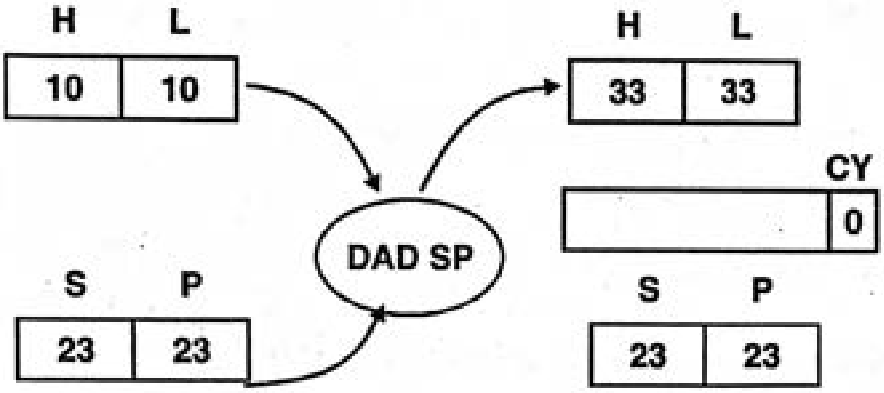


Fig. (2.8) Idea of DAD instructions

14) DAA-DECIMAL ADJUST ACCUMULATOR

This is a special arithmetic instruction. It adjusts binary contents of accumulator to two 4-bit BCD digits. This is the only instruction which uses Auxiliary carry (AC) flag for code adjustment. The CPU uses the flag internally. All flags are modified to reflect the result of execution. The adjustment process is as follows. The important condition is that this instruction must always follow an addition instruction for two BCD numbers. Thus it adjusts the sum of two BCD numbers to BCD and does not convert a binary number to BCD. Also it cannot be used to adjust results after subtraction.

The adjustment procedure is as follows

1) Carry out BCD addition – previous instruction.
2) If only lower nibble of accumulator is greater than 9 or if AC flag is set add 06 to lower nibble.
3) If only higher nibble of accumulator is greater than 9 or if C flag is set add 06 to upper nibble.
4) If both upper and lower nibbles are greater than 9 or AC and C flags are set respectively then add 66 to the accumulator contents.

Opcode	Operand	Bytes	Machine cycles	T-states
DAA	-	1	1	4

no emplicit operand.

Example:

a) Let us add 12 BCD to 39 BCD

	39 BCD	=	0011	1001
+	12 BCD	=	0001	0010
	51 BCD	=	0100	1011

The binary quantity (sum) is 4 BH We views that conditions 1 and 2 applies so add 06 to lower nibble.

$$\begin{array}{rlcll} & 4B & = & 0100 & 1011 \\ + & 06 & = & 0000 & 0110 \\ \hline & 51 & = & 0101 & 0001 \end{array}$$

Thus accumulator contents are adjusted to BCD value.

b) Let us add 68 BCD to 85 BCD

$$\begin{array}{rlcll} & 68\ BCD & = & 0110 & 1000 \\ + & 85\ BCD & = & 1000 & 0101 \\ \hline & 153\ BCD & = & 1110 & 1101 \\ & & & E & D \end{array}$$

In the accumulator binary sum is EDH. We view that conditions 1 and 4 are fulfilled so we add 66 H to EDH.

$$\begin{array}{rlcll} & EDH & = & 1110 & 1101 \\ + & 66\ H & = & 0110 & 0110 \\ \hline & 153\ BCD & = & 10101 & 0011 \\ & & & 1 \quad 5 & 3\ BCD \end{array}$$

Thus accumulator contents are adjusted to BCD.

This completes the arithmetic group instructions.

2.7 ILLUSTRATIONS AND PROGRAMMING EXERCISES

Three number addition: Write an assembly language program to add three numbers stored in consecutive memory locations starting from 1005 H and store the sum in next memory location.

Program:

```
Label    Mnemonic    Operand       Comments
         LXI H,      1005 H     ;  set HL pointer
         MOV         A, M       ;  load first number in Acc
         INX         H          ;  increment HL
         ADD         M          ;  add second number to first
         INX         H          ;  increment HL
         ADD         M          ;  add third number
         INX         H          ;  increment HL
         MOV         M, A       ;  store sum in memory
         HLT         -          ;  stop processing
```

EXERCISES

1) Write a program to subtract 16-bit binary number stored in B-C pair from 16-bit binary number stored in D-E pair store the result in H-L register pair.
2) Write a program to subtract 23 H from 7EH. Store the result at a memory location 712FH.
3) Write a program to add a byte data inputted from port 29H to the contents of accumulator and output the results on port 39H.
4) Write a program to add a 16-bit number stored in memory locations 1600H and 1601H (with lower byte first) to 16 bit number stored in DE pair. Store the results in HL register pair.
5) Write a program to add 13 BCD to 29 BCD store the results in L register (in BCD form).

2.8 LOGICAL GROUP INSTRUCTIONS

As seen earlier these instructions allow logical operations to be carried out. Logical operations such as ANDing, ORing, XORing, inverting etc. are allowed in 8085.

Following is detailed explanation for each of these instructions.

1) ANA-LOGICAL AND WITH ACCUMULATOR

This is a single byte instruction, which carries out logical ANDing of the contents of operand with the contents of Accumulator. The result is stored in accumulator itself.

Logical operations on 8-bit quantities are defined as 8 operations on respective bits. These 8 operations are independent.

S, Z, P flags are modified to reflect the result carry flag is reset and AC flag in set.

Opcode	Operand	Bytes	Machine cycles	T-states
ANA	reg	1	1	4
ANA	M	1	2	7

reg: any of the following registers B, C, D, E, H, L, A
M: memory location pointed by HL pair.

Example:

The contents of accumulator and the register D are 54 H and 82 H, respectively. If we execute the instruction

ANA D

The accumulator contents will be as follows:

```
          A    = 54 H   0101 0100
    AND
          D    = 82 H   1000 0010
   ------------------------------------
Acc  =    A.D  =        0000  0000 = 00H
```

Thus accumulator contents after ANDing are 00H. A_0 is ANDed with D_0 and so on upto A_7 ANDed with D_7.

This instruction is used to mask required bits during programming.

2) ORA-LOGICALLY OR WITH ACCUMULATOR

This single byte instruction carries out logical ORing of the contents of operand and contents of accumulator. The result is stored in accumulator itself. S, Z, and P flags are modified according to the result and AC and CY flags are reset.

Opcode	Operand	Bytes	Machine cycles	T-states
ORA	reg	1	1	4
ORA	M	1	2	7

reg.: any of the following registers B, C, ,D E, H, L, A

M: memory location pointed by HL pair.

Example: Consider ORA B

Contents before execution	Contents after execution
A BAH	A BBH
B 11H	B 11H
	flags S=1, Z=0, P=1, AC=0, CY=0

BA H = 1011 1010

11H = 0001 0001

ORAB = 1011 1011= BBH   flag = 1000 0100 = 84H

3) XRA- EXCLUSIVE-OR WITH ACCUMULATOR

This is a one-byte instruction. It makes EX-Oring the operand contents with Accumulator. The result is stored in Accumulator S, Z, P flags are modified according to result. CY and AC flags are reset.

Opcode	Operand	Bytes	Machine cycles	T-states
XRA	reg	1	1	4
XRA	M	1	2	7

reg.: any of the following registers B, C, D, E, H, L, A

M: memory location pointed by HL pair

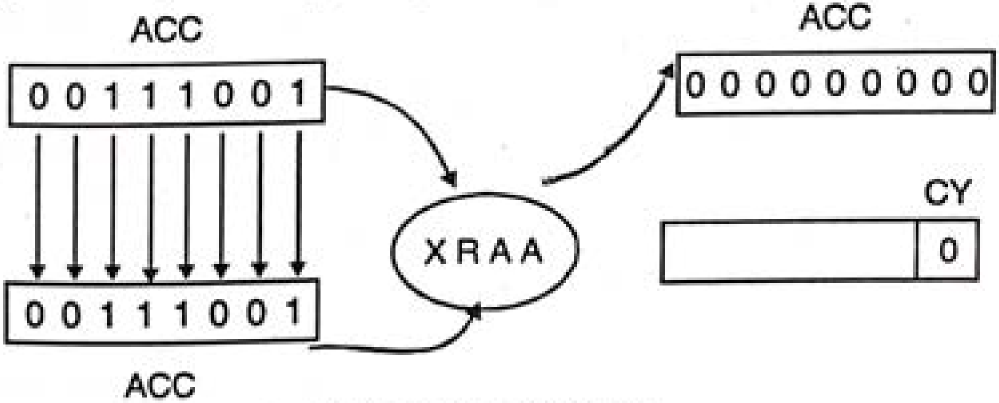


Fig. (2.9) Idea Of XRA A

.**Example:** i) Consider XRAB

Contents before execution	Contents after execution
A AA H	A FF H
B 55H	H 55H
	flags S=1, Z=0, P=1, AC=0, CY=0

AA H = 1010 1010
55H = 0101 0101
XRAB = 1111 1111= FFH flag = 1000 0100 = 84H

ii) **Consider XRAA as shown in fig.(2.9) if A= 39H then after XRAA it makes accumulator clear**

3) CMP-COMPARE WITH ACCUMULATOR

This is a single byte instruction. This instruction compares the contents of the operand (reg/memory) with the contents of accumulator. Both contents are preserved and a result of comparison is shown by setting the flags as below.

a) If [A] < [reg/memory] then **Carry** flag is **set.**
b) If [A] = [reg/memory] then **Zero** flag is **set.**
c) If [A] > [reg/memory] then both **Carry** and **Zero** flags are **reset.**

Note: [A] means contents of accumulator thus "[ ]" sign indicates "contents of ". In addition to Z and carry the other flags S, P, AC are also modified.

Opcode	Operand	Bytes	Machine cycles	T-states
CMP	reg	1	1	4
CMP	M	1	2	7

reg.: any of the following registers B, C,D E, H, L, A
M: memory location pointed by HL pair.

Example: Consider the instruction CMP L
We will view different cases for different values of A and L

		After execution		
Case No.	**Contents of A**	**Contents of L**	**Carry flag**	**Zero flag**
1	15 H	57 H	1	0
2	25 H	25 H	0	1
3	35 H	17 H	0	0

4) ANI-AND IMMEDIATE WITH ACCUMULATOR

This is a two byte instruction. It uses immediate addressing mode. The second byte acts as an 8-bit immediate data. It is logically ANDed with the contents of accumulator. The result is stored in accumulator S, Z and P flags are modified to reflect the result in the accumulator. Carry flag is reset, AC flag is set.

Opcode	Operand	Bytes	Machine cycles	T-states
ANI	data	2	2	7

data: 8-bit immediate data.

Example: The instruction ANI 3FH when executed, will store the result of ANDing 3FH with accumulator in accumulator.

5) ORI-OR IMMEDIATE WITH ACCUMULATOR

This also is a two byte instruction. It uses immediate addressing mode. The second byte is immediate 8-bit data. This data is logically ORed with the contents of the accumulator. The result is stored in the accumulator itself. S, Z, and P flags are modified to reflect the result in accumulator. Carry and AC flags are reset.

Opcode	Operand	Bytes	Machine cycles	T-states
ORI	data	2	2	7

data: 8-bit immediate data.

Example: The instruction ORI4CH when executed, will OR the contents of accumulator with 4CH and will store the result in accumulator.

6) XRI-XOR IMMEDIATE WITH ACCUMULATOR

This is again a 2-byte instruction using immediate addressing mode. The second byte which is immediate data is XORed with accumulator contents and result is stored in accumulator itself S, Z and P flags are modified to reflect the result in accumulator, carry and AC flags are reset.

Opcode	Operand	Bytes	Machine cycles	T-states
XRI	data	2	2	7

data: 8-bit immediate data.

Example: The execution of the instruction XRI FFH will store the result in accumulator, the result obtained by XORing accumulator contents with FFH.

7) CPI-COMPARE IMMEDIATE WITH ACCUMULATOR

This instruction is a two byte instruction utilizing immediate addressing mode. The execution of this instruction is very much similar to CMP instruction. The only difference being that in CPI instruction the immediate data is compared with accumulator.

S, P, AC flags are also modified in addition to Z and carry flag which reflect the result of comparison.

Opcode	Operand	Bytes	Machine cycles	T-states
CPI	data	2	2	7

8) CMA-COMPLEMENT THE ACCUMULATOR

This is a single byte instruction. When executed this instruction complements the accumulator contents. The result is stored in accumulator itself. No flags are affected.

Opcode	Operand	Bytes	Machine cycles	T-states
CMA	-	1	1	4

Examples: a) Consider the accumulator containing 57 H

A = 0101 0111

On execution of the instruction CMA the accumulator contents will change to

A = 1010 1000

i.e. A = A8 H

b) Write a program to find 1's complement of the number stored in memory location 65B0 H and store the result in memory location 6651H.

Program:

Label	Mnemonic	Opreand	Comments
	LDA	65B0 H	; Load data in Acc
	CMA	-	; Complement Acc
	STA	6651 H	; Store result in memory
	HLT	-	; Stop processing

9) CMC-COMPLEMENT CARRY

This also is a single byte instruction. When executed it complements the carry flag. No other flags are affected.

Opcode	Operand	Bytes	Machine cycles	T-states
CMC	-	1	1	4

10) STC-SET CARRY

This is also a one-byte instruction. When executed it sets the carry flag to 1. No other flags are affected.

Opcode	Operand	Bytes	Machine cycles	T-states
STC	-	1	1	4

11) RLC-ROTATE ACCUMULATOR LEFT

This is a single byte instruction. It is used to rotate each binary bit in accumulator to left by one position. Bit D_7 is placed in bit position D_0 as well as in the carry flag.

No flags other than carry flag are affected.

Opcode	Operand	Bytes	Machine cycles	T-states
RLC	-	1	1	4

This operation can be represented as follows:

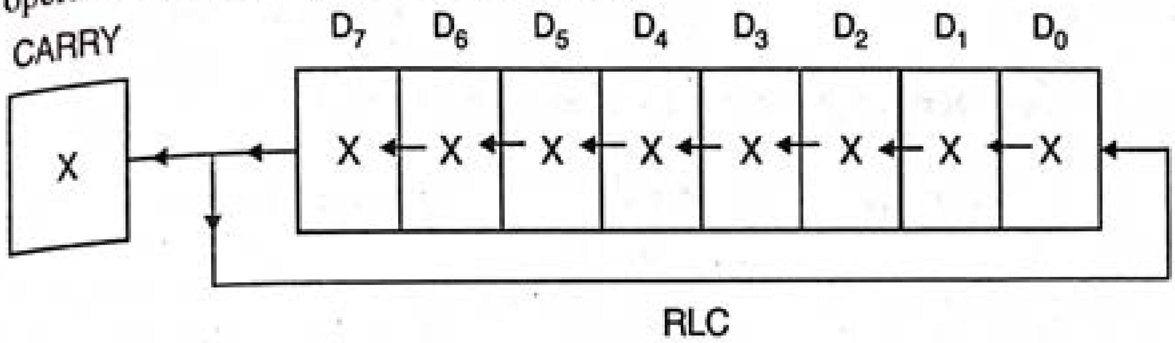


Fig. (2.10) RLC INSTRUCTION

Example:

Let contents of Accumulator be 83 H Let Carry = 0

	D_7							D_0
0	1	0	0	0	0	0	1	1

Carry

If we execute the instruction RLC then the accumulator and carry will be as follows:

	D_7							D_0
1	0	0	0	0	0	1	1	1

Carry Thus A = 07 H and Carry = 1

12) RRC-ROTATE ACCUMULATOR RIGHT

This is also a single byte rotate instruction. It is used to rotate each binary bit in accumulator to right by 1 position. Bit D_0 is placed in bit position D_7 as well as in carry flag. No other flags are modified.

Opcode	Operand	Bytes	Machine cycles	T-states
RRC	-	1	1	4

Example: This operation can be represented as follows

Let contents of accumulator be 83 H. Let carry be 0

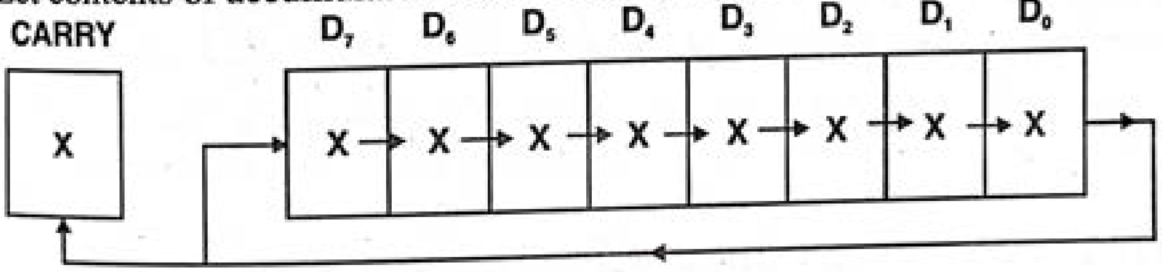


Fig. (2.11) RRC INSTRUCTION

Carry	D_7							D_0
0	1	0	0	0	0	0	1	1

If we execute the instruction RRC then contents of accumulator and carry will be as follows

	D_7							D_0
1	1	1	0	0	0	0	0	1

Carry Thus A = C1 H Carry = 1

13) RAL-ROTATE ACCUMULATOR LEFT THROUGH CARRY

This instruction is also a one byte instruction. It rotates ech binary bit of accumulator left by one bit position through carry flag. No other flags are modified. D7 is moved to carry and carry is moved to D0.

Opcode	Operand	Bytes	Machine cycles	T-states
RAL	-	1	1	4

This instruction can be represented as follows:

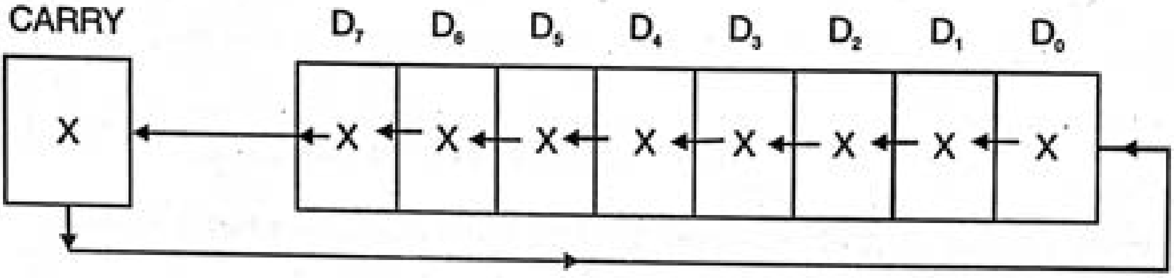


Fig. (2.12) RAL **INSTRUCTION**

Example:

Let contents of accumulator be 83 H. Let carry be 0.

Carry		D_7							D_0
0		1	0	0	0	0	0	1	1

After execution of the RAL instruction.

We get following in Accumulator and carry flag.

Carry		D_7							D_0
1		0	0	0	0	0	1	1	0

i.e. A = 06 H and Carry = 1

14) RAR-ROTATE ACCUMULATOR RIGHT THROUGH CARRY

This is an instruction, which rotates the accumulator contents to 1 bit position right the carry i.e.

D_0 goes to carry and carry goes to D_0. No other flags are affected.

Opcode	-	Operand	Bytes	Machine Cycles	T-states
RAR	-		1	1	4

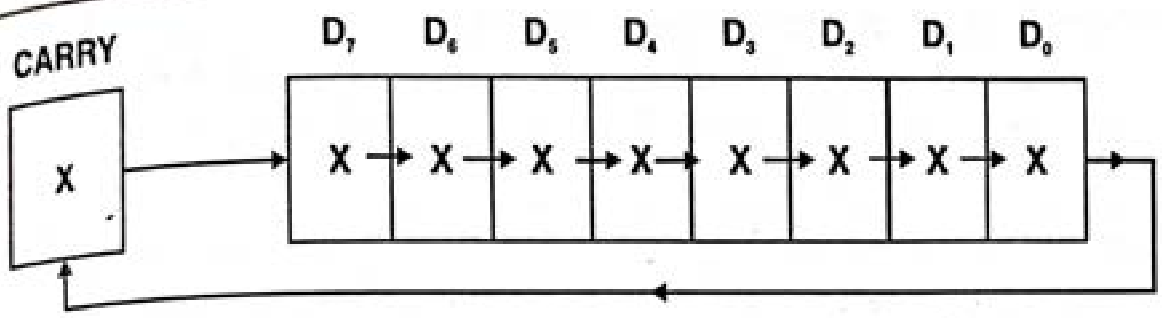


Fig. (2.13) RAR INSTRUCTION

This instruction can be represented as follows.

Example :

Let A = 83 H Carry = 0

	Carry	D_7 D_0
	0	1 0 0 0 0 0 1 1
On execution of RAR		
	Carry	D_7 D_0
	1	0 1 0 0 0 0 0 1

A = 41 H Carry = 1

This completes the discussion of Logical group instructions.

These instructions CMC, STC, CMA are sometimes called special instructions.

2.9 BRANCHING CONTROL GROUP OF INSTRUCTIONS – JUMP INSTRUCTIONS

As we have seen conditional jump and unconditional jump are the branching instructions. These are as follows.

1) UNCONDITIONAL JUMP – JMP

This instruction is a three-byte instruction. First byte is opcode, second byte is lower order address for branching and the third byte gives higher order address byte for branching. When this instruction is executed the program control is unconditionally transferred to the branch address given in instruction. Such an instruction allows user to set up unconditional loops. No flags are affected.

Opcode	Operand	Bytes	Machine Cycles	T-states
JMP	addr	3	3	10

Addr. 16 bit branch address

Example

The instruction JMP 3500H when executed will transfer the program control to the memory location 3500 H.

2) CONDITIONAL JUMP INSTRUCTIONS

These are also 3 byte Jump instructions. The second and third byte gives branching address with same significance as that for in conditional jump instruction. The only difference between conditional and unconditional branch is that conditional branch instruction first checks a certain condition. If condition is true then only branching takes place else program continues in same sequence. No flags are affected. Following are the conditional jump instruction.

Opcode	Description	Required status of flag for jump to be taken
JC	Jump on Carry	CY = 1
JNC	Jump on No Carry	CY = 0
JP	Jump on Positive	S = 0
JM	Jump on Minus	S = 1
JPE	Jump on parity even	P = 1
JPO	Jump on odd parity	P = 0
JZ	Jump on Zero	Z = 1
JNZ	Jump on no Zero	Z = 0

For all these instructions operand is 16-bit branch address.

2.10 EXERCISES

1) Write a program which will read 10 bytes from port 7EH one after and will output the sum of these bytes on port 8EH.

2) Write a program, which will fill memory locations 2900H to 29FFH with the data present at input port 77H.

3) Write a program to find how many memory locations from 3900H to 39FFH contain 00H. The number should be stored at the memory location 4000H.

4) Write a subroutine which will wait till MSB of the data inputed from port 11H gives high. On MSB going high it should output a FFH at port 12H.

5) Write a subroutine to multiply an 8-bit number stored at 1500H with 12H and store the result in HL register pair.

2.11 STACK

The stack in an 8085 based microprocessor system can be described as a set of R/W memory locations, specified by the program in the main program. These memory locations are used to store binary information (bytes) temporarily during the execution of a program.

The beginning of the stack is defined in the program by using a LXI SP instruction, which loads 16-bit memory address in stack pointer register of the microprocessor. In 8085 when we initialize stack pointer then storing of data bytes begins from a location with a address one less than the SP contents. And as we go on storing more and more data SP goes decrementing. Thus the stack grows from higher address to lower address. Therefore normally while writing 8085 programs users initialize SP at the highest available user R/W memory locations.

The size of stack is thus only limited by the available memory.

In 8085 we have following instruction to store the data and retrieve it back from stack.

a)	PUSH	To store operand contents on stack.
b)	POP	To load from stack into operand.

We will discuss these instructions in detail in forthcoming section.

One of the important uses of stack is done while executing a subroutine, But what exactly is a subroutine? Let us now see what does a subroutine exactly means and how it is executed.

2.12 SUBROUTINE

A subroutine is a group of instructions written separately from the main program to perform a function that occurs repeatedly in the main program. To avoid repetition of the same delay instructions, the subroutine technique is used. Delay instructions are written once only separate from the main program. The main program when required then calls these.

The 8085 microprocessor has two types of instructions to implement subroutines.

a) CALL instructions: These are conditional and unconditional calls. These instructions are used to call a subroutine.

b) RETurn instructions: These are conditional and unconditional returns. These instructions are used to return from a subroutine to main program.

The call instruction is written in the main program at a location where you wish to execute the subroutine. The return instruction is written at the end of the subroutine indicating end of subroutine.

When CALL instruction is executed the contents of program counter i.e. address of instruction following CALL are stored in the stack and the program control is transferred to subroutine. On completing the execution of subroutine the RET instruction is executed which loads back the stack contents i.e. address of the instruction following CALL instruction, in program counter. Thus the main program execution is resumed.

In the next section we will see subroutine control instructions in the branch control group.

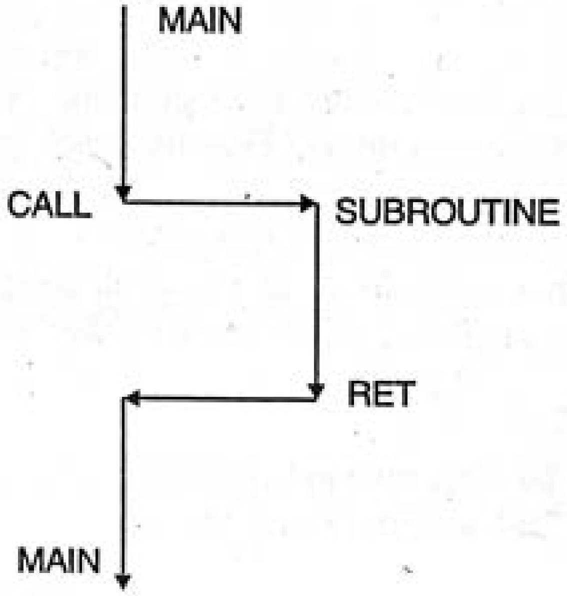


Fig. (2.14) Idea of Subroutine

2.13 BRANCHING CONTROL GROUP INSTRUCTIONS

1) UNCONDITIONAL CALL – CALL

This is a 3-byte instruction that transfers the program sequence to a subroutine address. First byte is opcode and second and third bytes give the subroutine address. On execution the address of the instruction following CALL is stored in stack. It decrements SP by two. It then jumps unconditionally to the subroutine address. This instruction is accompanied by a return type instruction at the end of subroutine. No flags are affected.

Opcode	Operand	Bytes
CALL	addr	3

addr : 16 bit subroutine address

Example :

Consider the following program sequence starting from memory 2000 H. Let there be a subroutine from 3000 H

Main	Subroutine
2000 CALL 3000 H	3000 MOV C,A
2003 MOV A,C	3001 IN 23 H
	3003 RET

On executing CALL 3000 H the address of the MOV A,C instruction i.e. 2003 H is stored in stack and program control transfers to address 3000 H. On executing RET, stack contents are loaded in PC and main program execution resumes.

2) CONDITIONAL CALL INSTRUCTIONS

These are also 3-byte call instructions but are conditional meaning that they check a certain condition and the subroutine is called only when the condition is true. No flags are affected.

Opcode	Description	Required status of flags for Call to be executed
CC	Call on Carry	CY=1
CNC	Call on No Carry	CY=0
CP	Call on Positive	S=0
CM	Call on Minus	S=1
CPE	Call on Parity Even	P=1
CPO	Call on Odd Parity	P=0
CZ	Call on Zero	Z=1
CNZ	Call on no Zero	Z=0

3)UNCONDITIONAL RETURN - RET

As seen earlier a return instruction indicates end of subroutine and transfers the control back to main program. This is a single byte instruction. On execution it loads two byte from stack i.e. address of instruction next to CALL into PC and increments SP by two. No flags are affected.

Opcode	Operand	Bytes	Machine Cycles	T-states
RET	---	1	3	10

3) CONDITIONAL RETURN INSTRUCTIONS

These are also one-byte instructions indicating end of subroutine on checking a certain condition and finding it true. Thus the control is transferred to main program in the same manner as for unconditional return only when condition is true else the subroutine execution is maintained. No flags are affected.

Opcode	Description	Required status of flags for Call to be executed
RC	Return on Carry	CY=1
RNC	Return on No Carry	CY=0
RP	Return on Positive	S=0
RM	Return on Minus	S=1
RPE	Return on Parity Even	P=1
RPE	Return on Odd Parity	P=0
RZ	Return on Zero	Z=1
RNZ	Return on no Zero	Z=0

4) RESTART INSTRUCTIONS – RST

We have seen the use and functioning of interrupts. We know that on giving the 8085 an interruption the pin INTR (Pin No.10) the microprocessor fetches the Interrupt service Routine (ISR) branch address from data bus. For this we should externally provide the instruction. RST instructions are one of the instructions, which can be used to transfer the control to ISR. These are 8 different RST instructions RST0 to RST7. These are all one byte instruction. In execution they are similar to call instruction with a predefined subroutine address obtained from the number associated with RST.

As a rule if n is the number associated with RST then the subroutine or ISR address is 8n. No flags are affected.

These instructions can be used in software to generate interrupts thus these are also called software interrupts.

Opcode	Operand	Bytes	Machine Cycles	T-states
RST	n	1	3	12

n = any no. from 0 to 7.This completes the subroutine control instructions.

2.14 MACHINE CONTROL GROUP OF INSTRUCTIONS

These operations include

a) Stack Operations.

b) Interrupt Control Operations.

c) Machine Control General Operations Such as Halting CPU etc.

A) STACK OPERATION INSTRUCTIONS

We have seen the use of stack, we now see the instructions, which allow one to store data on stack and retrieve it back. We will also see some data transfer instructions involving stack pointer (SP).

1) PUSH - PUSH REGISTER PAIR ON STACK

This is a single byte instruction. The contents of the register pair specified in the operand are copied into the stack in the following sequence.

a) The stack pointer is decremented and the contents of higher order register in pair (such as B in BC pair, D in DE pair) are copied on stack.

b) The stack pointer is decremented again and contents of lower order register are copied on the stack. No flags are modified Contents of register pair are unchanged.

Opcode	Operand	Bytes	Machine Cycles	T-states
PUSH	rp	1	3	12

rp : register pair any one of the following.

rp	**Pair Name**	
	High Byte	Low Byte
B	B	C
D	D	E
H	H	L
PSW	A	F

A : Accumulator F : Flag register PSW : Program status word.

Example : PUSH D

Will push contents of DE pair

Let D = 15 H & E = 23 H Let SP = 2300 H

Then after executing PUSH D we will get following contents in SP and stack.

SP = 22FE	STACK	
22FE	23H	← SP
22FF	15H	
2300		

Fig. (2.15) Push Instructions

2) POP - POP OFF STACK TO REGISTER PAIR

This is a single byte instruction. On execution copies two top bytes on stack to designated register pair in operand. The execution follows the sequence given below.

a) Contents of top most location of stack called stack top are copied into lower register (such as C in BC etc) of the pair. The SP is incremented by 1.

b) Contents of the stack location pointed by SP are copied into higher register of the pair. The stack pointer SP is incremented by 1.

No flags are affected. Contents of stack are unchanged.

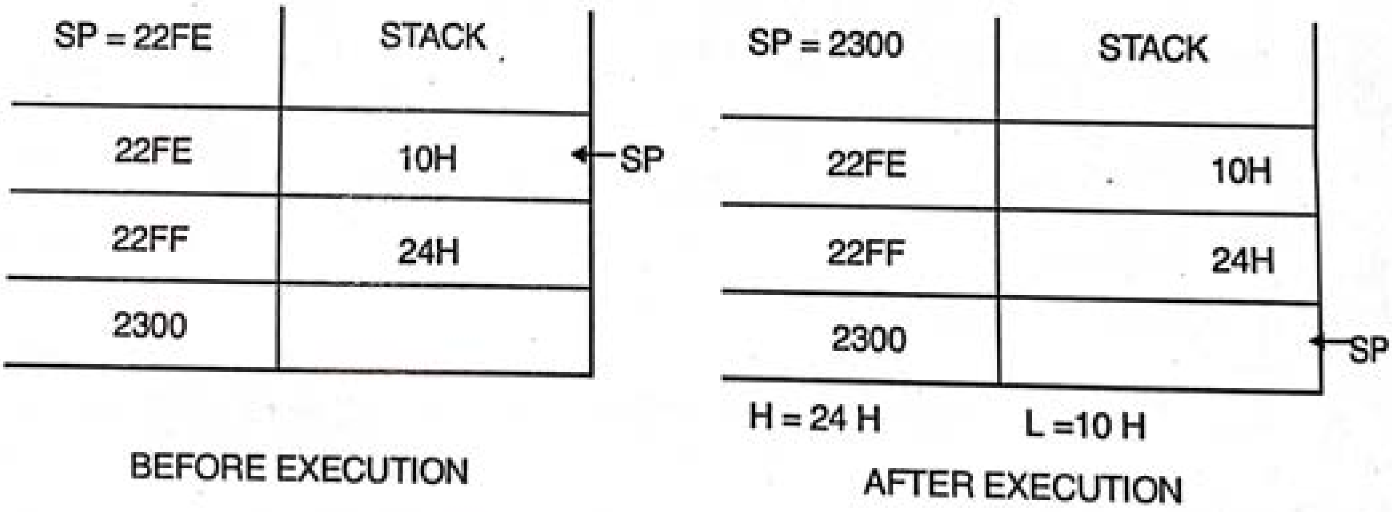
Opcode	Operand	Bytes	Machine Cycles	T-states
POP	rp	1	3	10

rp : Same as rp for PUSH instruction.

Example :

Consider SP = 22FE H with following contents stored on stack.

On execution of instruction POP H the contents of H, L, SP will be as shown in fig.(2.16)



SP = 22FE	STACK	
22FE	10H	←SP
22FF	24H	
2300		

BEFORE EXECUTION

SP = 2300	STACK	
22FE	10H	
22FF	24H	
2300		←SP

H = 24 H L =10 H

AFTER EXECUTION

Fig. (2.16) POP Instruction

3) XTHL – EXCHANGE H AND L WITH TOP OF STACK

This is a single byte instruction. On execution the contents of L register are exchange with contents of stack top byte i.e. Contents at memory location pointed be SP. Also contents of H register are exchanged with the contents of memory location pointed by SP + 1. Contents of SP are not altered. No flags are affected.

Opcode	Operand	Bytes	Machine Cycles	T-states
XTHL	--	1	5	16

Example :

Consider following stack contents and HL pair contents.

Stack
H = 26 H L = 15 H AB H SP
CDH

On execution of the instruction XTHL the contents of HL pair and stack are as follows.

Stack
H = CD H L = ABH 15 H SP
26 H

4) SPHL – COPY H AND L REGISTER TO SP

This is a single byte instruction which copies contents of L register into lower byte of SP and contents of H register into higher byte of SP. The contents of H and L registers are unaffected. No flags are affected.

Opcode	Operand	Bytes	Machine Cycles	T-states
SPHL	--	1	1	6

B) INTERRUPT CONTROL OPERATIONS

These instructions allow masking, enabling and disabling of interrupts. In addition the masking instruction allow serial I/O to be carried out. Following are the instructions in this group.

I) SIM – SET INTERRUPT MASK

This is a single byte multipurpose instruction it allows setting of masking of interrupts and serial output. It uses accumulator contents for this purpose. Accumulator contents are interpreted as follows :

D7	D6	D5	D4	D3	D2	D1	D0
SOD	SDE	X	R7.5	MSE	M7.5	M6.5	M5.5
Serial	Serial data		Reset	Mask	Mask respective interrupt		
Output	enable		Int 7.5	enable if bit set to 1			
Data	1 = enable		flipflop if 1		0 = disable		

Opcode	Operand	Bytes	Machine Cycles	T-states
SIM	--	1	1	4

2) RIM – READ INTERRUPT MASK

This also is a single byte instruction, which is used for reading interrupt masking information as well as serial data input. It utilizes accumulator contents for that purpose. The significance of accumulator contents is as follows.

D7	D6	D5	D4	D3	D2	D1	D0
SID	17	16	15	IE	7.5	6.5	5.5
Serial Output Data	Bit=1 indicates pending interrupt			Interrupts enabled if 1	1=interrupt masked 0=interrupt not masked		

On execution the contents of accumulator reflect respective conditions as above.

Opcode	Operand	Bytes	Machine Cycles	T-states
RIM	--	1	1	4

3) EI – ENABLE INTERRUPT INSTRUCTION

This is a single byte instruction. On execution interrupt enable flip flop is set and all interrupts are enabled. No flags are affected.

Opcode	Operand	Bytes	Machine Cycles	T-states
EI	--	1	1	4

4) GENERAL MACHINE CONTROL OPERATIONS.

These are for CPU halting, no operation etc. Following are the instructions.

1) PCHL – LOAD PROGRAM COUNTER WITH HL REGISTER PAIR CONTENTS.

This is a single byte instruction. It copies contents of HL pair into PC. The result is equivalent is to 1-byte unconditional jump with address stored in HL pair. No flags are affected.

Opcode	Operand	Bytes	Machine Cycles	T-states
PCHL	--	1	1	4

2) NOP – NO OPERATION

This is a single byte instruction. On execution no operation is performed only instruction is fetched and decoded. No flags are affected. It can be create time delays using loops.

3) HLT - HALT AND ENTER WAIT STATE

This is a single byte instruction. After completing its execution CPU goes to a halt state halting further execution. Wait states are inserted in every clock cycle. During Halt CPU maintains register contents. But tri-states address and data lines. An interrupt or reset is necessary to exit from Halt state.

This completes machine control group instructions.

SOLVED PROGRAMS

1) Write a program segment to add three byte integer in register EHL from another three byte integer in BCD. The result should be placed in BCD registers keeping the integer in EHL undisturbed. **(March 88 – 5 Marks)**

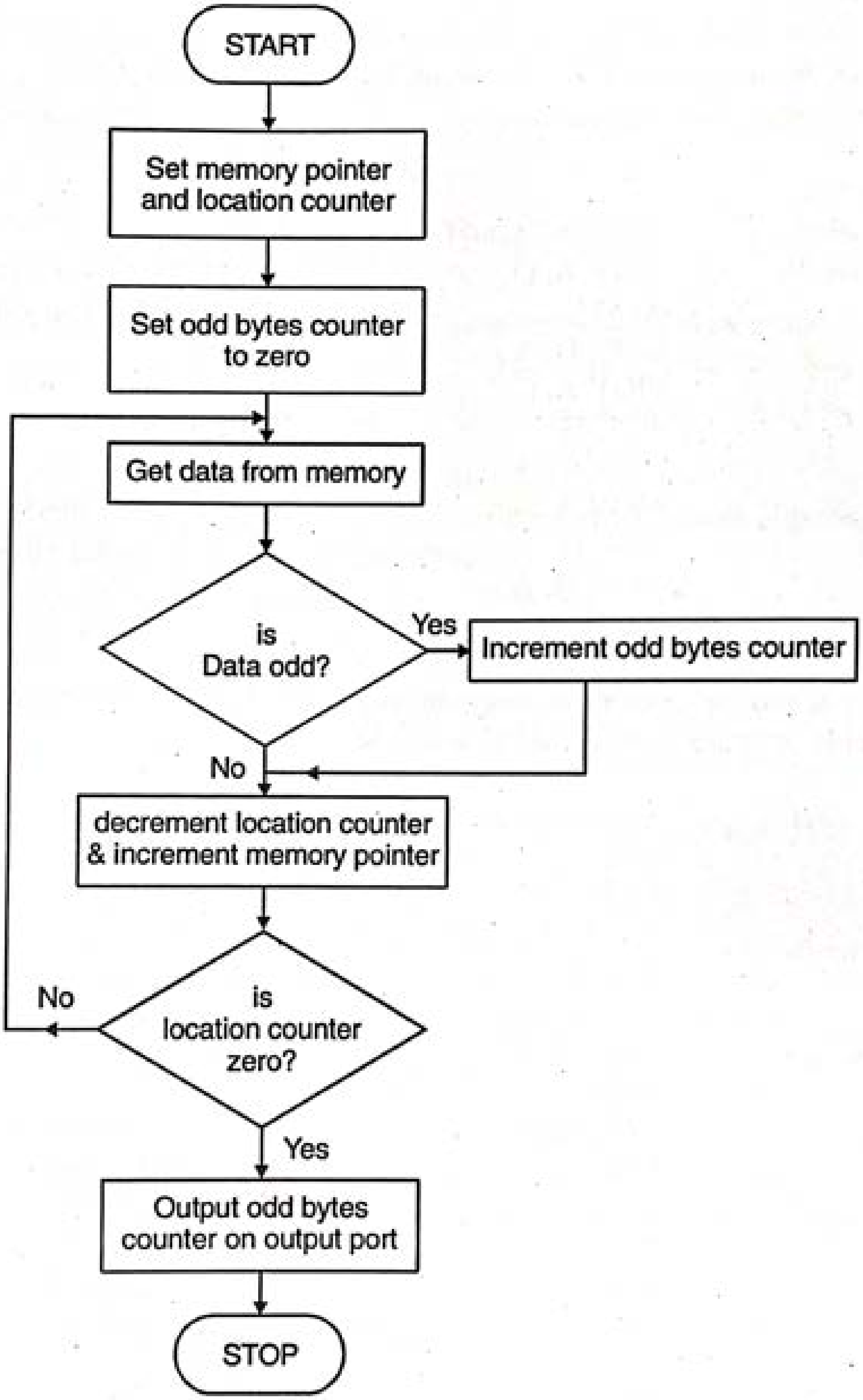
Solution :

```
Label     Mnemonic and Operand     Comments
AAD3      MOV A, D                 least significant byte
          ADD L                    add without carry (D + L)
          MOV D, A                 result in D
          MOV A, C                 middle byte
          ADC H                    add with carry
          MOV C, A
          MOV A, B                 most significant byte
          ADC E                    add with carry (B + E)
          MOV B, A
          HLT
```

2) Write a program segment to count number of odd data bytes in the block of memory from 1300 H to 13FFH and output on port 11H. **(5 Marks)**

```
Label     Mnemonic and Operand     Comments
          STC
          CMC                      ; Clear carry
          LXI H, 1300 H            ; Set HL pointer to 1300H
          MVI B, FFH               ; Set location counter to FFH
          MVI C,00H                ; Set odd bytes counter to 00H
Again     MOV A,M                  ; get data from memory.
          RAR
          JNC Skip                 Check odd or even
          INR C                    if odd byte counter increments
Skip      INXH                     increment memory pointer
          DCR B
          JNZ Again                Checks if 255 bytes are checked.
          MOV A,M                  Loads 256th byte if 255 bytes are
                                   checked
                                   Checks odd or even
          JNC Next                 If odd, increments odd
          INR C                    byte counter and copies it to
Next      MOV A,C                  accumulator which is then output
          OUT 11H                  to output port 11H.
```

Flowchart:


START
Set memory pointer and location counter
Set odd bytes counter to zero
Get data from memory
is Data odd?
Yes
Increment odd bytes counter
No
decrement location counter & increment memory pointer
is location counter zero?
No
Yes
Output odd bytes counter on output port
STOP

3) Compare the contents of BC register pair with that of DE pair send data AAH on output port B if they are equal otherwise status of the port should be kept unchanged. Write a subroutine to achieve above task. Assume the port B is initialized for output. **(Oct 89 - 5 Marks)**

Solution

```
Label     Mnemonic and Operand     Comments
Start     XRA A                    ; Clear Acc
          MOV A,C                  ; Copy C to accumulator
          CMP E                    ; Compare acc.with E reg.
          JNZ Finish               ; If not equal end the routine
          MOV A,B                  : Copy B to accumulator
          CMP D                    ; Compare acc with D reg.
          JNZ finish               ; if not equal end the routine
          MVIA, AAH                ; Load A with AAH
          OUT PB                   ; Output AAH on port B as BC
                                     pair
                                   ; contents equals DE pair
                                   contents
Finish    RET                      ; Return from subroutine.
```

4) Write a program segment using appropriate 'rotate' instruction to divide the number in BC register pair by 2. The quotient should remain in BC register pair. **(Oct 88 – 5 Marks)**

Solution :

```
          Label     Mnemonic and Operand     Comments
                    STC
                    CMC                      ; Set carry to 0
                    MOV A,B
                    RRC                      ; carry to MSB and LSB of
higher
                                              order
                    MOV B,A
                    RRC                      ;reminder to carry
                    MOV C,A
```

Verification :

If B = 0110 1000 = 68 H
C = 0010 0100 = 24 H
After RRC B = 0011 0100 (68÷2=34H)
C = 0001 0010 = 12H (24÷2=12H)

5) The accumulator contains the data 76 H and the register L contains the data A6H. What will be the contents of accumulator in hex after execution of each of the following instruction independently.

i) ORAL ii) ANA L iii) RRC iv) XRA L v) ADDL

(March 89 – 3 Marks)

i) ORAL
A = 76 H = 0111 0110
L = A6H = 1010 0110

A + L = F6H = 1111 0110

Thus after execution A = F6 H

ii) ANA L
A = 76 H = 0111 0110
L = A6H = 1010 0110

A . L = 26H = 0010 0110

Thus after execution A = 26 H

iii) RRC
A = 76 H = 0111 0110
After RRC
A = 0011 1011

3BH and carry = 0

iv) XRAL
A = 76 H = 0111 0110
L = A6H = 1010 0110

A ⊕ L = DO H = 1101 0000

Thus after execution A = DO H

v) ADDL
A = 76 H = 0111 0110
L = A6H = 1010 0110

A + L = 1C H = 10001 1100 Carry = 1
Thus after execution A = 1C H Carry = 1

6)Multiply the 16 bit unsigned number in memory location 3340 H by the 8-bit unsigned number in memory location 3341H.Place the result in memory location 3342H and 3343H such that 8 LSB's are in 3342H and 8 MSB's are in 3343H.

(March 89 – 5 Marks)

Solution :

```
Label   Mnemonic and Operand        Comments
                LHLD 3340 H         ; get first byte in H and second in L
                XCHG                ; get first byte in D and second in E
                LXIH, 0000H                 ; clear HL for result
                MOV B, D                    ; get first byte in B as counter
                MVI D,00H                   ; clear D
Repeat          DAD, D                      ; add second byte to result
                DCR B                       ; decrement counter
                JNZ Repeat                  ; if counter is zero
                SHLD 3342H                  ; HL contains results
```

7) Write a program segment to wait till at least one line of input port A to be high. Then continuously output square wave of arbitrary frequency on all lines of port B. (assume port A and port B already configured for input and output respectively) **(March 88)**

```
Label           Mnemonic and Operand    Comments
SQR             IN PA                   ; read port A
                ADI 00H                 ; modify flags accordingly
                JZ SQR                  ; if [A] = 00 then wait
                OUTPB                   ; if [A] = 00 output data on port B
Again           CMA                     ; output on port B
                JMP Again               ; continue output
```

8)Write a subroutine to fill the memory locations 2800H to 28FFH with Hex numbers 00H to FFH respectively (i.e.2800 H will contain 00H,2801H will contain 01H etc.)

(March 88 – 5 Marks)

Solution :

We will solve this problem in two manners.

Approach 1:

```
Label           Mnemonic and Operand    Comments
Fill            LXI H, 2800H            ;set memory pointer
                MVI A,00H               ;load Acc with 0
                STC
                CMC                     ; Clear carry
BBK             MOV M, A                ; Store Acc to memory
                INX H                   ; increment momory pointer
                INR A                   ; increment data
                JNC BBK                 ; go back if not over
                RET                     ; return if over
```

Approach 2:

```
Label     Mnemonic and Operand     Comments
Fill      LXI D, 2800H             ; set memory start address.
          XRA A                    ; clear Acc & carry
          MVI B, FFH               ; load counter
Loop      STAX D                   ; store data in memory
          INR A                    ; increment data
          INX D                    ; increment memory address
          JNC Loop                 ; go back if not over
          RET                      ; return if over
```

Important Note: (You can use any valid instructions, any program is valid if it is written by using correct instructions without changing the purpose of the program).

9) Write a program segment to count number of odd data bytes in the block of memory from 1300 H to 13FFH and output on port 11H. **(5 Marks)**

```
Label     Mnemonic and Operand     Comments
          STC
          CMC                      ; Clear carry
          LXI H, 1300 H            ; Set HL pointer to 1300H
          MVI B, FFH               ; Set location counter to FFH
          MVI C,00H                ; Set odd bytes counter to 00H
Again     MOV A,M                  ; get data from memory.
          RAR
          JNC Skip                 Check odd or even
          INR C                    if odd byte counter increments
Skip      INXH                     increment memory pointer
          DCR B
          JNZ Again                Checks if 255 bytes are checked.
          MOV A,M                  Loads 256th byte if 255 bytes are
                                   checked
          JNC Next                 Checks odd or even
          INR C                    If odd, increments odd
Next      MOU A,C                  byte counter and copies it to
          OUT 11H                  accumulator which is then output
                                   to output port 11H.
```

10) Write a program to count how many times 05 comes in a memory block atarting at 4000H to 4004H. Store the result at 4070H.

Solution :

```
Lebel  Mnemonics       Comments

       LXI H 4000H     Load the HL pair at 4000H.

       MVI C, 00H      Clear the C register for counter.
```

```
        MVI D, 05H      Store the total No. 05 in D register
Start   MVI A, 05H      Store the no. 05  in A register.
        MOV B, M        Transfer the contents of memory to B register.
        SUB B           Subtract the value of B register from A.
        JNZ Loop        If A ≠0 then go to Loop.
        INR C           Otherwise increment the counter by 1.
Loop    DCR D           Decrement the D register by 1.
        JZ end          If D = 0 then go to end.
        INZ H           Increment the HL pair by 1.
        JMP start       Again go to start.
        MOV A,C         Transfer contents of register C to acc.
        STA 4070H       Store result at 4070H
End     RSTI            Stop
```

11)Write a program to chock whether data 02H is present in a memory block starting at 4065H to 4070H. If present store 'FF'H at 4080H. Else store '00'H at the same location.

Solution :

```
Lebel   Mnemonics       Comments
        LXI H 4065H     Load the HL pair at 4065H.
        MOV B,M         Move the contents of memory to B.
        INX H           Increment the HL Pair by 1.
        MVI C,O         Initialize C with 0.
        LDA 4050H       Load the accumulator at 4050H
Loop    CMP M                   Compare memory with A
        JZ Done         jump if zero to Done.
        JC Note 1       jump if carry to Note 1.
        INX H           Increment the HL pair by 1.
        DCR B           Decrement B register by 1.
        JNZ Loop        Jump if not zero to Loop.
```

```
Note 1      MVI C,FFH       Move FF to C.
Done        MOV A,C,        Move C to Accumulator
            STA 4050H       Move the contents of Accumulator to memory
                            4050H
End         RST 1           Stop.
```

12)The accumulator contains the data 77H and register B contains 66H. What will be contents of accumulator in hex digits after execution of each of the following instruction?

(i) ANI 07H (ii) XRI 11 H (iii) CMA (iv) ORAB (v) RLC

Solution : (i) Accumulator contents in BCD (A) = 77H = 0111 0111
Register B contents in BCD (B) = 66H = 0110 0110
Therefore ANI 07H = in hex, 07H

(ii) XRI 11H – XOR accumulator immediate with 11 H.

A = 0111 0111 BCD

11H = 0001 0001 BCD

A XOR 11H =0110 0110 BCD = 66H

Therefore XRI 11H = in hex, 66H

(iii) CMA – complement the accumulator contents

A = 0111 0111 BCD

CMA = 1000 1000 BCD = 88H

Therefore CMA = in hex, 88H

(iv) ORAB – OR accumulator with contents of register B.

A = 0111 0111 BCD

B = 0110 0110 BCD

A + 11 = 0111 0111 BCD = 77H

Therefore ORAB = in hex, 77H

(v) RLC – Rotate accumulator left through carry.
. In hex after execution of RLC A = 1110 1110 = EEH

14) Write an assembly language program to count number of even data bytes occurring in a block stored from memory location 3001H and onwards. The length of block is stored in location 3000H. Store the result in location 3100H.

(March 2002)

```
Label    Mnemonic/operand     Comments
START    LXI H, 3000H ;       Initialize H-L pointer to memory address 3000H
         MOV C, M ;           Get length of block in register C
         MVI B, OOH ;         Initialize register B to store count
         INX H ;              Increment H-L pair by 1
UP       MOV A, M ;           Get the number in accumulator
         RRC ;                Check even number
         JC DOWN ;            Jump on carry to label DOWN
         INR B ;              No carry – increment count
DOWN     INX H ;              Increment H-L pair by 1
         DCR C ;              Decrement content of C by 1
         JNZ UP ;             Is zero ? No – jump to label UP
         MOV A, B ;           Store count in register A
         STA 3100 H ;         Store the count of even byte in location 3100H
END      HLT                  Stop processing
```

15) A Hex number is stored at location 3000H. Write an assembly language program to interchange it's digits. The new number is to be stored at 3001H.Add original number with new number and store the result at location 3010H. **(March 2002)**

```
Label    Mnemonic/operand     Comments
START    LXI H, 3000H ;       Initialize H-L pointer to memory location 3000 H
         MOV A, M;            Get the number in accumulator
         RRC ;  }
         RRC ;  }             Exchange two nibbles
         RRC ;  }
         RRC ;  }
         STA 3001H ;          Store the exchanged number in memory location
                              3001 H
         ADD M ;              Add two numbers
         STA 3010 H ;         Store the result in 3010 H location

END      HLT                  Stop processing
```

16) Write an assembly language program to count the number of times the data ADH is found in a block of memory locations starting from 3000H. Length of block is stored in location 29FF H. Store the result in location 2000H. (March 2002)

```
Label     Mnemonic/operand     Comments
START     MVIC, OOH ;          Initialize register C
          MVI A, ADH ;         Get the data ADH in accumulator
          LXI H, 2FFFH ;       Initialize H-L pointer to 2FFFH
          MOV B, M ;           Get count in register H
          INX H ;              Increment H-L pair
LOOP      CMP M ;              Compare the number
          JNZ NEXT ;           Is zero ? No – jump to label NEXT
          INR C ;              Increment content of C
NEXT      INX H ;              Increment H-L pair by 1
          DCR B ;              Decrement count
          JNZ LOOP ;           Is zero ? No – jump to label LOOP
          MOV A, C ;           Yes – move count in A
          STA 2000H ;          Store count in 2000 H
END       HLT                  Stop processing
```

16) Write the appropriate memories for the following tasks. **(March 97)**

i) load accumulator from B register

ii) Complement the accumulator iii)Add 01 H with the accumulator

iv) Store the contents of accumulator at the memory location addressed by the BC register pain v)Clear the accumulator.

Solution :i) MOV A,B ii) CMA iii)ADIA 01H iv) STAXB v) ANA 00H

QUESTIONS

1) Select correct alternative and rewrite the following:

1) The flag bit that gets affected by execution of RRC instruction in 8085 processor is ________. **(Oct. 88)**

i) zero, ii) parity, iii) carry, iv) all

2) The instruction that can affect stack pointer is ________.

i) SHLD, ii) XCHG, iii) LXI, iv) LDAX **(Mar. 88)**

3) Which of the following instruction does not affect the flag.

i) PCHL, ii) ADD, iii) RAR, iv) STC **(Mar. 89)**

4) PUSH (rp) instruction in 8085 affect ________.

i) HL register, ii) Program Counter, iii) stack Pointer, iv) None of these

5) ________ is the three byte instruction. **(Mar. 99)**

i)RAR, ii) MOV A,D, iii) LXI SP 2050H, iv) SBI 80H

6) The invalid register pair for 8085 microprocessor is ___________. **(Mar.2002)**

i) BC ii)HL iii)SP iv)DE

7) In 8085 microprocessor, flag register is not affected after the execution of _____ instruction. **(Mar.2004)**

i) INR r, ii) DCR r, iii) ADD r, iv) INX rp

8) The full form of instruction DAA is ________________. **(Mar.2005)**

a) Double Add Accumulator b) Decimal Add Accumulator

c) double Adjust Accumulator d) Decimal Adjust Accumulator

9) _______ instruction rotates the contents of ACC left through carry by 1 bit.

a) RLC b) RRC c) RAR d) RAL **(Mar.2006)**

10) The instruction which affects only carry flag is __________. **(Mar.2007)**

a) ORI b) XRI c) ADI d) DAD

11)____________ instruction uses flags. **(Mar.2007)**

a) Data Transfer b) Arithmetical c) Conditional Jump d) Logical

12) The instruction ________________ will affect the zero flag without changing the contents of the accumulator. **(Mar.2008)**

a) MVI A, 00 b) SUB A c) XRA A d) CMP A

13)—— of the following instructions is a branching instruction. **(Mar.2010)**

a) ADD r b) JMP addr c) CMP M d) CMP A

14) ________________ instruction does not affect the flags. **(Mar.2016)**

a) RAR b) CMPC c) XRA d) MOV A,B

15)________________Instruction is used for 16-bit addition. **(Mar.2016)**

a) ADD b) ADI c) ADC d) DAD

16) ANA , r instruction comes under_____ group.

a) Arithmetic b) Logical c) Branch d) Data Transfer

THEORY QUESTIONS

1) Define the following terms with suitable diagram: **(Mar.2002,05,10,17,19)**

a) Instruction cycle b)Machine cycle c) T-state

2) Explain the following instructions with suitable example: **(Mar. 2017)**

i)LDAX, ii) XCHG iii) DAD

3) The flag register of 8085 microprocessor contains the data 45H. Interpret it's meaning. **(Specimen Paper)**

4) Explain the following instructions with suitable example: **(Oct. 03)**

i)DAA, ii) LHLD

5) Explain the following instructions of 8085 microprocessor with suitable example of each: **(Mar.2004)**

i) SHLD addr, ii) ANI data, iii) RRC

6) Explain the following instruction of 8085 micro-processor. **(Mar.2005)**

i) CMC ii) RAR iii) XRA M

7) If ACC contains data BCH, register C contains ADH. What will be the content of accumulator after execution of each of the following instructions independently?

i) SUB C ii) CMA iii) XRA C **(Mar.2006)**

8) Accumulator of 8085 contains data 56 H. What will be the contents after the execution of following instructions independently. **(Mar.2005)**

i) CMA ii) ANI ACH iii) INR A

9) Explain the execution of following instructions of 8085 micro-processor with suitable examples. i) CMP C ii) DAD rp **(Mar.2005)**

10) Explain the following Intel 8085 instructions: **(Mar.2007)**

i) DAA ii) NOP iii) LDAX D

11) Explain the difference between SUB and CMP instruction. **(Mar.2007)**

12) Explain conditional and unconditional branching instructions with suitable examples. List any two machine control group of instructions with suitable example. **(Mar.2006)**

13) Explain the following instructions of 8085 microprocessor with suitable example: a) SPHL b)PCHL **(Specimen Paper)**

14) Write any three instructions to make Accumulator clear or zero. **(Mar.2017).**

15) Explain INX r instruction **(Mar.2019).**

PROGRAMMING EXERCISE

1) Write an assembly language program to find out 2's compliment of five numbers stored from memory location 3330H and onwards. Store the result from memory location address 410H. **(Oct. 03)**

2) Write an assembly language program to count the occurrence of the data 9CH in a memory block starting from 4000H to 400FH. Store the count at memory location 4500H **(Mar.2004)**

3) Write an assembly language program to transfer a memory block starting from 1050H to1059H a new block starting from 1070H to 1079H. **(Mar.2016)**

4) Write an assembly language program to add 2 decimal numbers stored at 1050H and 1051H store the result at 1052H and 1053H. **(Mar.2016)**

5) Write an assembly language program to perform the multiplication of two 8-bit numbers where multiplicand id stored at the memory locations 2501H and 2502H and multiplier is stored at 2503H. The result is to be stored at memory location address 2504H to 2505H. **Oct.03,Mar.2016)**
(Note : **8 bit multiplication is extended to 16 bit)**

6) Write a subroutine to rotate 16-bit HL register pair, one binary place towards left. (The MSB of H should shift to LSB of L). **(Mar. 88)**

7) The accumulator contains the data AAH and register B contains 55H. What will be the contents of accumulator in hex digits after execution of each of the following instructions?
i) ANAB, ii) XRAB, iii) ORAB, iv) ANI81H, v) CMA

8) The Accumulator of 8085 contains data 43H. What will be it's contents after the execution of following instructions independently ? **(Oct. 2003)**
i) CMA, ii) ANT 09H, iii) INR A

9) Write an assembly language program, which tests all 1's condition of data-byte specified at data memory location 3500H. If all bits are 1's, store 01H at 3501H else store 00H at the same location. **(Oct. 90)**

10) The length of the block is in memory location 2040H and the block itself begins from location 2041H. Write a program to find largest element in the block and store the result at location 2280H. **(Mar. 91)**

11) The accumulator contains the data 77H and register B contains 66H. What will be the contents of accumulator in hex digits after execution of each of the following instructions?
i) ANI 07H, ii) XRI 11H, iii) ORAB, iv) RLC, v) CMA **(Mar. 91)**

12) Write an assembly language program to count the number of times the data ADH is found in a block of memory locations starting from 3000H. Length of block is stored in location 2FFF H. Store the result in location 2000H **(Mar.2002)**

13) The accumulator of 8085 microprocessor contains the data 45H and register 'E' contains the data 7BH. What will be the content of accumulator after execution of each of following instructions independently?
i) XRA E, ii) ADI C5H, iii) ORI 5BH **(Mar.2004)**

14) Write an assembly language program to increment the contents of alternate memory locations each by two from 1051H to 1060H. **(Mar.2016)**

15)Write an Assembly language program to fill the memory locations 4500H to 4504 with the Hexadecimal numbers 09H to ODH respectively. **(Mar.2017)**

16) A block of data is stored from memory location D001H. Length of block is stored at D000H. Write a program to find occurrence of data 02H in given block. Store the number of occurrence at Memory Location D100H. **(Mar.2020)**

Answers Q. 1)

1)Carry 2) LXI 3) PCHL 4)Stack Pointer 5)LXI SP2050H 6)SP 7) INX rp.

8) Decimal Adjust Accumulator 9) RAL 10) DAD 11) Conditional Jump

12) CMP A 13) JMP addr 14) MOV A,B 15) DAD 16) Logical

□□□

3

INTRODUCTION TO INTEL X-86 FAMILY

INTRODUCTION

As explained in the last topic the first microprocessor was a 4-bit µp and then in 1978 Intel introduced the µp 8088 and 8086. The IBM-PC and PC-XT were based on 8088 and then Intel started this family by means of advance versions such as 80286,80386 ,80486 working as powerful microprocessors. The development of new CPU chips by Intel continued the latest microprocessors Pentium, PentiumII ,and Celeron core came in to existence. These microprocessors developed by Intel known as X-86 family. This chapter describes their features and brief information of 8086.

3.1 16-BIT MICROPROCESSORS

16-bit µps families are used primarily in µcomputers and are oriented towards high-level languages. They have powerful instruction sets and are capable of addressing megabytes of memory.

e.g. Intel 8086/8088
80186/286
Zilog Z 8001/8002
Motorola 68000 and NS 16000

Apart from the design concepts and instructions sets, one of the critical factors that decide capability of µp is the number of pins available.

Primary objectives of these X-86 family are:

1) Increase memory-addressing capacity.
2) Increase execution speed.
3) Provide a powerful instruction set.
4) Facilitating programming in high-level languages.
5) Function in multiprocessor environment.

These objectives can be met by using various design concepts.

Let us see each microprocessor with their special features

INTEL 8086/8088

- It is a16-bit µp
- 40 pin DIP package
- Capable of addressing 1 MB of memory its address bus is 20 bit (2^{20}=1MB)
- Clock frequencies from 5 MHz. to 10 MHz.
- 8-bit external data-bus.
- Operates in two modes Minimum mode and Maximum mode.

- 8088 and 8086 are identical in internal architecture, only differ in external design.
- Power and clock are identical to that of the 8085 processor.
- Signals for multi-processor environment.
- Data bus and status signals are multiplexed with the address bus.
- 16 data lines A_{D15}-A_{D0} multiplexed.

3.2 INTEL 8086 ARCHITECTURE

Fig.(3.1) shows an architecture of 8086 with two main processing units EU and BIU. When 8085 μp fetches and executes the instruction. The buses are occupied for the fetch operation but remains idle during internal execution of an instruction. To speed up the execution, 8086 processor includes two processing units called Execution Unit (EU) a Bus Interface Unit (BIU).The concept of dividing work between two units and processing it simultaneously speeds up the execution.BIU fetches the instructions and places them in instruction queue. EU continuously executes them until it becomes empty. Such a processing is known as parallel processing.

A segment register provides a base address and another register supplies offset address to increase memory addressing capacity.

EU and BIU of 8086

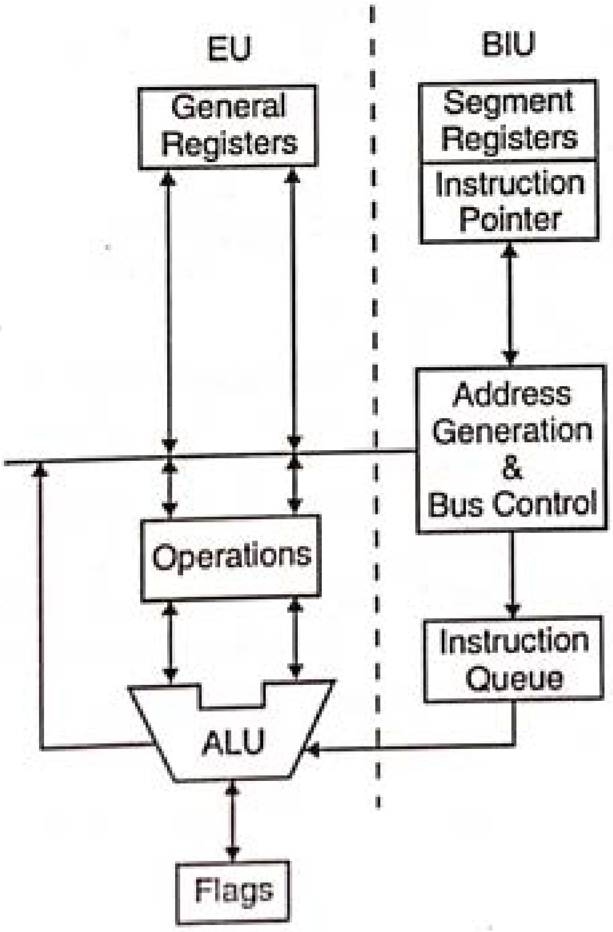
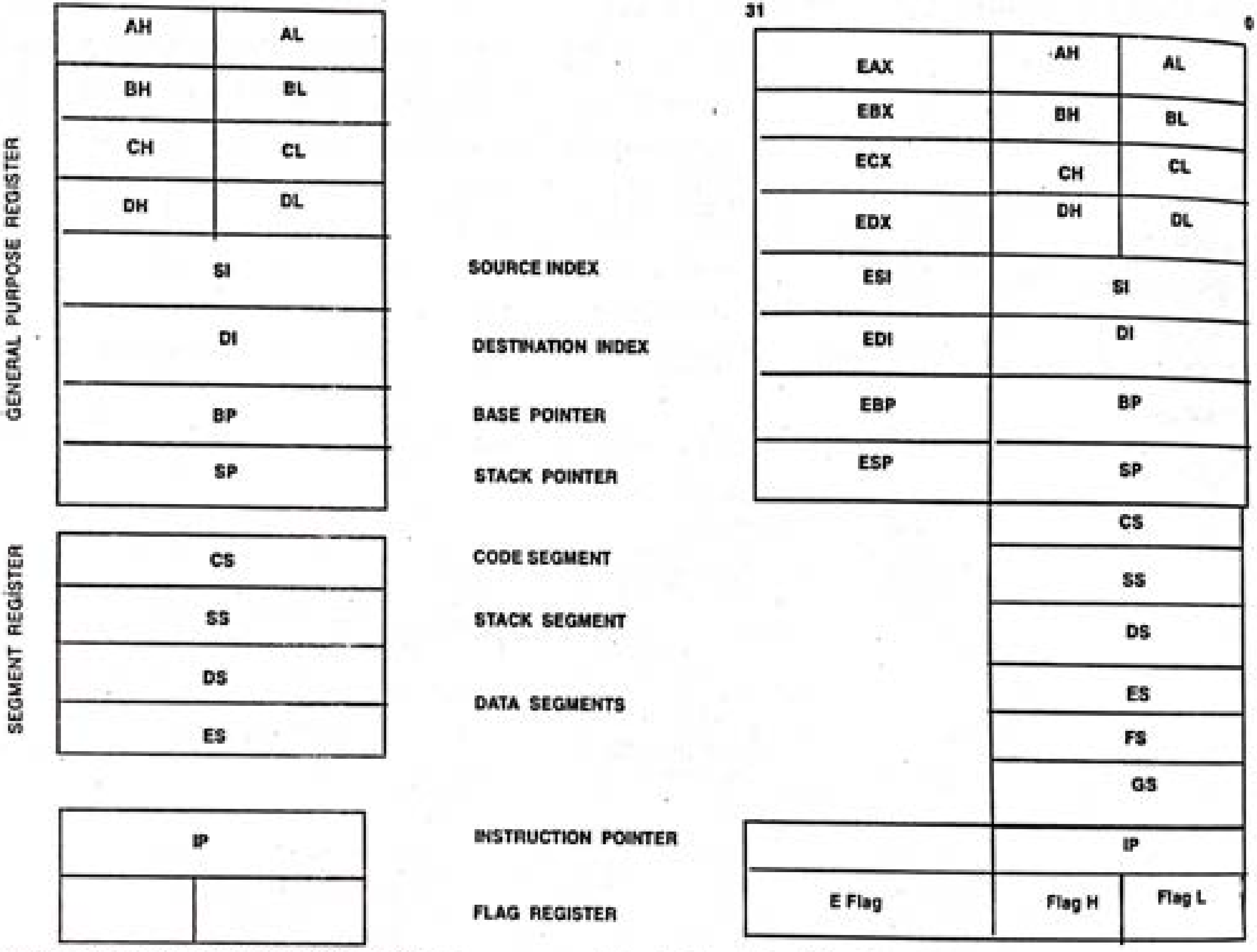

EU
BIU
General Registers
Segment Registers
Instruction Pointer
Address Generation & Bus Control
Operations
Instruction Queue
ALU
Flags

Fig.(3.1) Block Diagram of 8086

Programming Model Of 8086

The 8086 processor includes four 16-bit general-purpose registers: AX, BX, CX, DX. They are equivalent to 4 Accumulators even though some registers are assigned additional special functions. Each register can be associated with as 8-bit register and are compatible with 8085 registers .e.g. HL register pair in 8085 is same as BX register. BX: BH (B-Higher) and BL (B-Lower)


15
0
AH
AL
BH
BL
CH
CL
DH
DL
SI
DI
BP
SP
GENERAL PURPOSE REGISTER
CS
SS
DS
ES
SEGMENT REGISTER
IP
SOURCE INDEX
DESTINATION INDEX
BASE POINTER
STACK POINTER
CODE SEGMENT
STACK SEGMENT
DATA SEGMENTS
INSTRUCTION POINTER
FLAG REGISTER
31
0
EAX
EBX
ECX
EDX
ESI
EDI
EBP
ESP
FS
GS
E Flag
Flag H
Flag L

Register Model of 8088/8086/80286 Register Model of 32-bit µp386/486/pentium

Fig.(3.2) Programming Register Model Of X-86 family

The next 4 registers are SP, BP, SI and DI, which are 16-bit registers (used as memory pointers).

- SP (Stack Pointer), BP (base pointer), SI (Source Index) and DI (Destination Index) are used as index registers which are not present in 8085.
- Next group of registers are called as segment registers: CS (Code Segment), DS (Data Segment), SS (Stack Segment) and ES (Extra Segment).
- These segment registers are combined with memory pointers to generate a memory address.
- Instruction pointer (IP) is same as Program Counter in 8085.
- A flag register includes 9 flags. These flags are divided into two groups:

  6- bit data flags and 3- bit control flags.

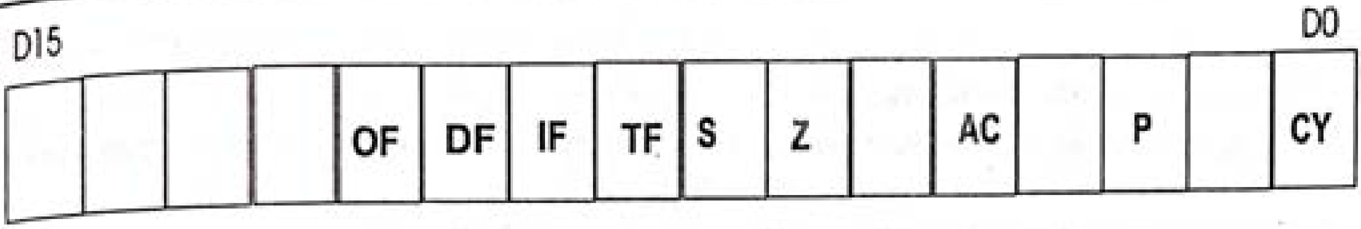


Fig.(3.3) Flag Register of 8086

The first five bits are identical to 8085.

In data flag category OF (Overflow) flag is there, rests of the flags are same as 8085 flags.

- OF – Overflow Flag – Used in signed numbers, when the result of signed number is too large, causing MSB to overflow into the sign bit, this flag is set.
- DF – Direction Flag – Used to control the direction (increment/decrement) of the string operation.
- IF – Interrupt Flag – Used to enable or disable external maskable interrupt requests.
- TF – Trap Flag – used for single stepping instructions.

Programming register model of 32-bit version of µp

As shown in fig. (3.2) the programming register model is almost identical with 8086 only the registers have Extended length hence called EAX,EBX,ECX, ESI, EDI etc. these registers can be used as 16-bit or 32-bit registers. The segment registers are identical with 8086 but additional flags (E flags) are used in these µp .The flag register is 32-bit.

Comparison of 8086 With 8085

Microprocessor 8085	Microprocessor 8086
1. It is a 8-bit Microprocessor.	1. It is a 16-bit Microprocessor.
2.Address bus is 16-bit hence can access 64 Kb memory	2.Address bus is 20- bit hence can access 1Mb memory
3.It provides 8-bit Registers A,B,H,...	3.It provides 16-bit Registers Ax,Bx,,...
4. 8-bits of Address bus $AD_0 - AD_7$ are multiplexed.	4. 16-bits of Address bus $AD_0 - AD_{15}$ are multiplexed.
5.Flag Register is 8-bit and contains 5 flags.	5.Flag Register is 16-bit and contains 9 flags.
6.Operated only in one mode	6. There are two operating modes.

INTEL 80286

- It is a 16-bit Microprocessor extended version of 8086.
- 68 pin DIP package
- Capable of addressing 16 MB of memory its address bus is 24bit (2^{24}=16MB)
- Clock frequencies from 6MHz. to 20 MHz.

- It can support a memory management unit, through it can address 1Gb memory known as virtual memory.
- This processor includes various built in mechanisms that can protect system software from user programs and restrict access to some regions of memory.
- It is designed for multiuser system.

INTEL 80386

- It is a 32-bit Microprocessor extended version of 80286.
- 132 pin grid array package
- Capable of addressing 4GB of memory its address bus is 32bit (2^{32}= 4GB)
- Clock frequencies from 16MHz. to 33 MHz.
- 80386 has 32-bit registers and compatible with 8086
- Execution of instructions is highly pipelined and the processor is designed to operate in a multiuser and multitasking environment.

Comparison of major features of X-86 family

Microprocessor	**8086**	**80286**	**80386**	**80486**	**Pentium**
Data bus (bits)	16	16	32	32	64
Address Bus (bits)	20	24	32	32	32
Operating speed MHz	5-10	6-20	16-33	25-50	50-100
Memory capacity	1MB	16 MB	4GB	4GB	4GB
Memory management	External	External	External	Internal	Internal
PC Type (IBM)	PC-XT	PC-AT	PC-AT	PC-AT	PC-AT
Math Co-Processor	External	External	External	Internal	Internal
Introduction	1978	1982	1985	1989	1993

INTEL 80486

- Upgraded, faster version of 80386.
- DX type version is a 32-bit processor housed in a 168-pin grid array package and can operate with clock frequencies from 25 MHz. to 66 MHz.
- Design is based on 1.2 million transistors compared to 300 thousand transistors of 386 processor.
- Important concepts that are built in to 486 other than 386 are:
  a) Built-in math-co-processor
  (In 386 math-co-processor is external)
  ∴Math instructions in 486 systems are executed three times faster than in 386.
  b) 8 Kbyte of code and data cache memory on chip.
  c) Highly pipelined execution unit.
  ∴The execution time for many instructions is 1 clock period.

- Generally used for high-end μcomputers and new environment.
- Designed to facilitate the execution of high-level languages and are suited for multiuser and multiprocessing systems.

3.3 INTEL PENTIUM PROCESSOR

- Pentium processor has a 32-bit address bus and 64-bit data bus and designed to operate from 60 MHz. to 233 MHz.
- Upward software compatible with the previous line of Intel μps from 8086/88 to 486 and it is also designed for easy upgradability.
- It includes many advanced features normally available in mainframe computers.
- The processor is ideally suited for high-end desktop PCs, workstations and R/W fileservers where high-speed computation like 3D graphics is needed.
- Can run an advanced operating system like UNIX.
- Includes enhancement of some of the features available on 486 processor. e.g. Supports either traditional memory page size 4Kbyte or the larger size of 4 Mbyte.
- Has 64-bit data bus, increases the processing speed.
- Suited for multiprocessing applications.
- Maintains data integrity in multiple μps.

ADVANCED DESIGN FEATURES OF PENTIUM

1) Dual Pipelining architecture

It has two execution units with dual pipelined architecture, able to execute two instructions simultaneously per clock cycle and achieve a high level of performance.

2) On-chip cache memory for instruction and data

It is found that the speed of CPU and main memory is mismatching main memory speed is very low as compared to CPU speed. To compensate this mismatch a high speed memory is used to store current programs and data and to make it available to CPU with faster rate. The cache holds those segments of program and data which are frequently required by CPU and for remaining program and data CPU access main memory. Using a cache by CPU results in best performance of a computer system. All latest CPU make use of cache. Pentium has 2, 8 Kbytes on cache memory on chip, one for code and other for data. Which eliminates the need for the processor to go off the chip and access the main memory during the loop execution, thus improving performance of μp.

3) The Branch prediction

This is the mainframe technique implemented in this μp to improve the performance. Here most likely set of instructions to be executed is predetermined. Using branch prediction it makes an intelligent guess of the next conditional instruction, it prevents the instruction cache from running dry during conditional instruction.

4) High performance floating point unit

This µp has an on-chip floating point unit that incorporates highly sophisticated 7-stage pipelining and hardwired codes.

6) Performance monitoring

Processor design enables the user to monitor the performance of the processor.

7) 64-bit data bus

Pentium has a wide 64-bit data bus which results in high speed data transfer.

QUESTIONS

1) Select the correct Alternatives
   a) The microprocessor 8088 is a ______ bit microprocessor.
      i) 8-bit, ii) 16-bit, iii) 64-bit
   b) µp 8086 has ______ flag register.
      i) 8-bit, ii) 16-bit, iii) 12-bit
   c) The first 32-bit microprocessor is ________.
      i) 8085, ii) 8086, iii) 80386, iv) 80486
   d) The only 64-bit microprocessor in X-86 family is _______.
      i) 80386, ii) 8086, iii) 80486, iv) Pentium
   e) 80386 is a _____. **(Mar. 2003)**
      a) 8 bit microprocessor, b) 16 bit microprocessor,
      c) 32 bit microprocessor, d) 4 bit microprocessor
2) Compare microprocessor 8085 with 8086.
3) Compare any four specifications of different µp in X-86 family.
4) Compare any four features of 80386 and 80486. **(Mar. 2017)**
5) What do you mean by: a) cache memory, b) multiplex address/data bus
6) Explain the programming model for 32 bit version of X-86 family with suitable diagram. **(Mar. 2002,05)**
7) Explain the advantages of the following features of the Pentium Processor.
   a) Dual Pipelining, b) On Chip caches, c) Branch Prediction,
   d) 64-bit Data Bus **(Specimen Paper,Mar.2006,10,16,20)**
8) List the advanced microprocessors of INTEL X-86 family and mention three attributes of anyone of them. **(Oct.2003)**
9) Compare any four attributes of 80286 and Pentium microprocessor. **(Mar. 2004)**
10) Explain the main features of a Pentium processor. **(Mar.2003)**
11) Draw a neat labeled diagram of flag register of X86 family. **(Mar.2003,16)**
12) Draw a neat labeled diagram of programming model of X86 family. **(Mar.07,08)**
13) Discuss in brief the members of X-86 family beginning from 80386. **(Mar. 2016)**
14) Compare any four features of 80486 and Pentium Processor **(Mar. 2019)**

Answers Q.(1)

(a)16-bit (b)16-bit (c)80386 (d)Pentium (e) 32 bit microprocessor

❑❑❑

4

INTRODUCTION TO MICROCONTROLLER

INTRODUCTION

The microcontroller is a complete microprocessor system built on a single integrated circuit. Generally building a complete microprocessor system on single chip reduces the cost of building simple products using microprocessor. When microprocessor system is implemented with a single chip microcontroller, it is also called microprocessor as all the major parts are in the IC.

It is generally used for control function so it is termed as microcontroller. Generally microcontroller must include a full implementation of a standard microprocessor, ROM, RAM, Parallel I/O port timers and a clock.

Single chip 4-bit microcontrollers were available shortly after mid 1970's. Texas instruments introduced TM S-1000 series of single chip 4 bit microcontrollers in mid 1970's and at the same time Fairchild introduced 8 bit two chip system. 8 bit microcontrollers become available after 1977 and 1978.

4.1 ADVANTAGES OF MICROCONTROLLER

1. Since the microcontroller is a single chip microcomputer it can be used for dedicated functions.
2. It can be used as independent controllers in various machines.
3. It includes all the essentials of a computer on a single chip like CPU, R/W memory, ROM it can be used as a microcomputer
4. It reduces the cost of a system than microprocessor based system. So it can be used in low cost products like toys.

4.2 APPLICATIONS OF MICROCONTROLLER

1. Most personal computer keyboards are implemented with a microcontroller. It replaces scanning, debounce matrix decoding and serial transmission circuits.
2. Generally in low cost products, such as toys, electric drills, microwave ovens, VCRs microcontrollers are used.
3. Microcontrollers are used as machine tools, chemical processors and in medical instruments.
4. It also controls mechanism of electronic systems, music system, home security system, etc.

In this chapter we will study a brief overview of 8051 microcontrollers.

4.2 MICROCONTROLLER 8051

The 8051 is a second generation 8-bit microcontroller. The first Intel's 8-bit microcontroller was the 8048.The 8051 provide a more powerful architecture, a more powerful instruction set, a full serial port.

Main Features Of 8051 Microcontroller

1) An 8 bit ALU 2) 4K × 8 ROM (OR EPROM)
2) 128 × 8 RAM 4) Dual 16 bit timer event counter
5) 32 I/O lines
6) Addresses of 64 Kbytes of program memory
7) Addresses of 64 Kbytes of data memory
8) Powerful 111 instruction set
9) Full featured serial port
10) Up to 12 MHz. clock
11) Two external interrupts

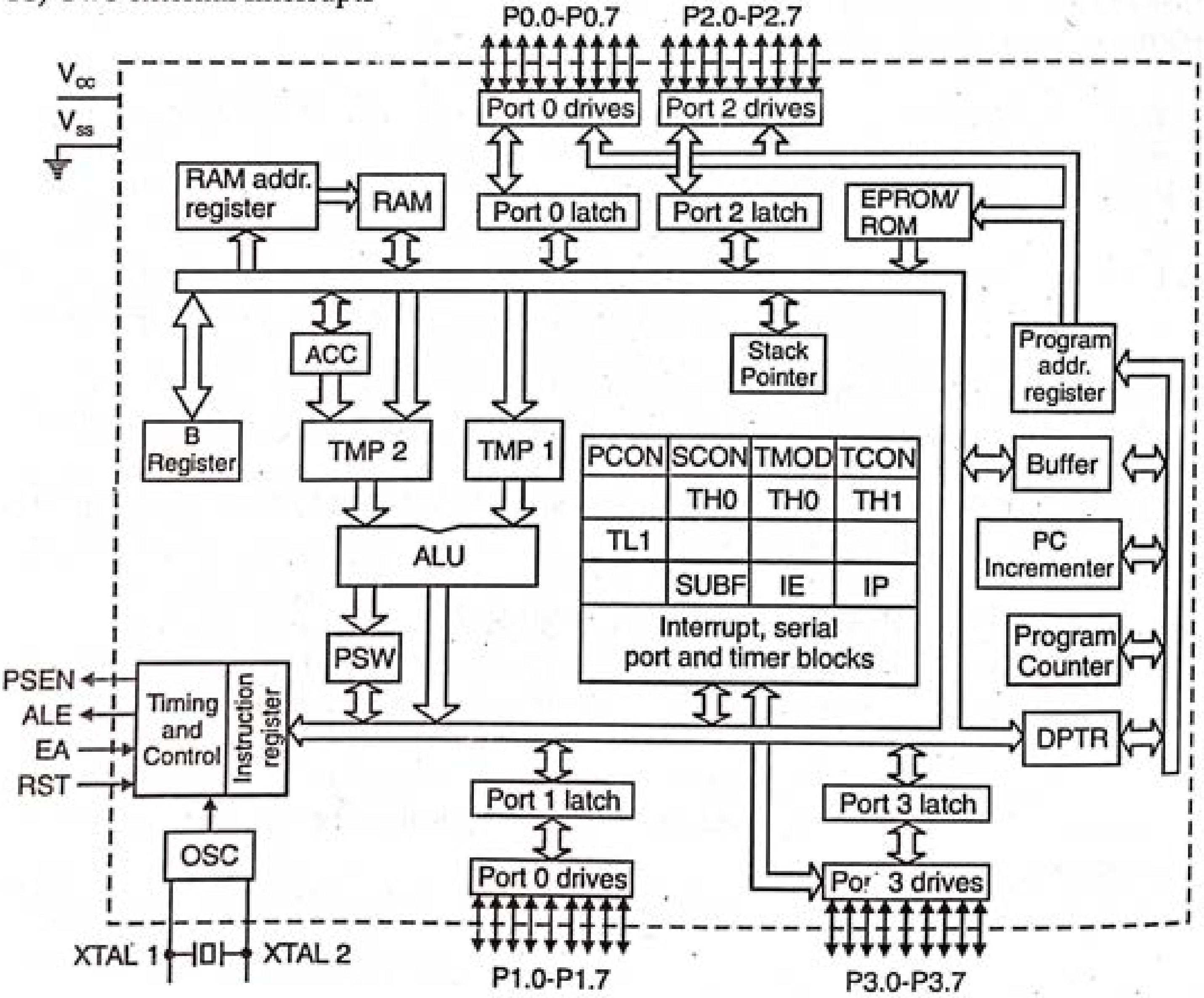

P0.0-P0.7
P2.0-P2.7
V_{cc}
V_{ss}
Port 0 drives
Port 2 drives
RAM addr. register
RAM
Port 0 latch
Port 2 latch
EPROM/ ROM
ACC
Stack Pointer
Program addr. register
B Register
TMP 2
TMP 1
PCON
SCON
TMOD
TCON
TH0
TH0
TH1
TL1
SUBF
IE
IP
Interrupt, serial port and timer blocks
Buffer
PC Incrementer
Program Counter
ALU
PSW
PSEN
ALE
EA
RST
Timing and Control
Instruction register
DPTR
Port 1 latch
Port 3 latch
OSC
Port 0 drives
Por 3 drives
XTAL 1
XTAL 2
P1.0-P1.7
P3.0-P3.7

Fig. (4.1) An architectural block diagram of 8051

Architecture Of 8051

There are 32 pins needed by the four 8-bit bi-directional I/O ports. Eight additional pins provide power and allow to connect to a clock crystal and also provide timing and control signals.

The standard functions, which make up a microprocessor, are in the center of the diagram. It includes the ALU, accumulator, stack pointer, a block of registers and a general purpose registers. All of these devices are connected to 8051 internal 8-bit data bus.

Each I/O port is also connected to the 8 bit internal data bus through a series of register. These registers hold data during I/O transfers and control I/O ports. It is also having ROM and RAM.

8051 Memory Register Map

Generally 8051 addresses two memory spaces. It uses one memory space for storing programs and the other for storing variable data. The program memory space is a read only memory space. You can read program instructions from this space but the processor cannot write data or read data from these memory locations. The 8051 internal ROM is in program memory space. All instructions fetches are from program memory space.

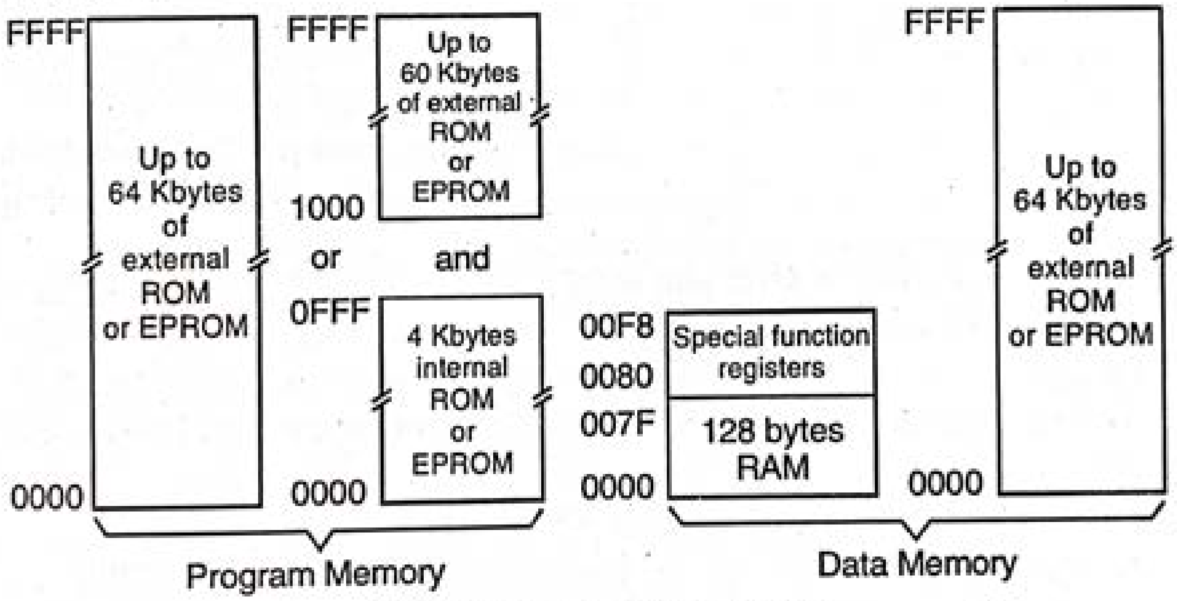

FFFF
Up to 64 Kbytes of external ROM or EPROM
0000
FFFF
Up to 60 Kbytes of external ROM or EPROM
1000
or
and
0FFF
4 Kbytes internal ROM or EPROM
0000
00F8
Special function registers
0080
007F
128 bytes RAM
0000
FFFF
Up to 64 Kbytes of external ROM or EPROM
0000
Program Memory
Data Memory

Fig.(4.2) Memory Map of 8051

The data memory space is read-write memory space. The processor can read data from this memory space and can write data to this memory space. But it cannot execute program instructions from this memory space. The 8051 internal RAM is in this memory space.

The 128 bytes of internal RAM provide general read-write data storage. Part of this memory space is often referred to as general purpose registers.

The 8051 also has 22 special function registers which are not part of 128 bytes of internal RAM. They occupy memory space from 80H to F8H.

If more program memory is needed the internal 4 Kbytes memory can be expanded by an additional 60 Kbytes. So 8051 has now a full 64 Kbytes program memory space. If 8051 pin is connected to ground, it does not use the internal 4 K ROM. If user needs more RAM, external data memory can be added. It is of 64 Kbytes. Generally 8051 operates with separate program memory and data memory space but in some applications it is desirable to have these work as common memory. In that case 8051 has 64 Kbytes of total external memory.

In this configuration 8051 can input a block of data through serial communication port, load data into memory and then execute that data as a program. This is called downloaded program.

Other Microcontrollers In The 8051 Family

8048 is the first Intel Microcontroller.

It is similar to 8051.

The architectural block diagram of 8048, 8049, 8050 all have similar architecture but memory size is different. It doubles in each case. 8048 supports 1 KB of internal memory 8049 support 2 KB and 8050 supports 4 KB.

These three (8048, 8049, 8050) are very popular as they are inexpensive.

8052 is simple expansion of 8051.

8050 has 8 Kbytes of on board ROM and 256 bytes of on board RAM. 8052 costs more than 8051. It has also one extra 16-bit counter timer, which gives programmer more flexibility. 8051 and 8052 has on board ROM, which must be programmed by device manufacturer. They are used generally in high volume applications, which can afford the cost of semiconductor operation.

For low volume and prototyping applications 8751 and 8752 are used. They uses on board EPROM. Using EPROM memory can be erased with ultraviolet light and that memory can reprogram.

Two alternate versions of 8051 and 8052 are 8031 and 8032. These devices do not have on board ROM so external ROM (OR EPROM) has to be used for program memory. Another form of 8052 is 8052 AHBASIC. This special 8052 has BASIC programming language in ROM. Using BASIC, programmer can write instruction for 8052.

Comparison with microprocessor

Microprocessor 8085	Microcontroller 8051
a) It is an 8-bit μp.	a) It is a 8-bit microcontroller.
b) Address bus is 16-bit, hence can access 64KB memory.	b) Address bus is 16-bit, hence can access 64KB memory
c) It provides seven 8-bit registers –A, B, H ...	c) It provides 34 8-bit registers – A, Band 32 general purpose registers.
d) 8-bit of data bus but ports are not available.	d) It has four ports P0-P3 for I/O
e) Flag register is 8-bit and contains Five flags.	e) Flag register is 8-bit and contains Nine flags.
f) Peripheral chips are required	f) Peripheral chips are not required

QUESTIONS

1) Select the correct alternative and rewrite the following:

a) The 8051 microcontroller has an instruction set of ______ instructions.
i) 99, ii) 111, iii) 120, iv) 110 **(Mar. 2002,05)**

b) In 8051 size of internal ROM is ______.
i) 4 KB, ii) 2 KB, iii) 8 KB, iv) 16 KB

c) In ______ microcontroller onboard EPROM is used.
i) 8751, ii) 8048, iii) 8052, iv) None of these

d) 8051 is ______ bit microcontroller.
i) 4 bit, ii) 8 bit, iii) 16 bit, iv) 32 bit

e) ______ is a microcontroller chip. **(Specimen Paper)**
i) 8085, ii) 8086, iii) 8051, iv) Pentium

f) _____ is not a characteristic feature of 8051 microcontroller. **(Mar. 2004)**
i) 4kbyte of internal RAM, ii) 4 kbyte of internal ROM
iii)4 parallel bidirectional I/O port iv)Full featured serial port

g) The 8051 microcontroller has an ALU of _____ bit capacity. **(Oct.03)**
i) 8, ii) 16, iii) 32, iv) 64

h)____________ IC consists internal RAM. **(Mar. 2006)**
a) 8080 b) 8085 c) 8051 d) 8086

i)Among following ________________ is the latest 8-bit single chip microcontroller.
a) 8048 b) 8051 c) 8096 d) 8044 **(Mar. 2007)**

j) ________________ is a Microcontroller. **(Mar. 2016)**
a) 8086 b) 8051 c) 8088 d) 80286

1) Differentiate between Micro-controller and Microprocessor. **(Mar. 2016)**
2) Define microcontroller. State any four advantages of the same over microprocessor-based system. **(Specimen Paper Mar.2002,05)**
3) Explain memory register map of 8051 with a suitable diagram. **(Mar. 2016,19,20)**
4) Explain various applications of microcontroller. **(Mar. 2002,10)**
5) Discuss the micro-controllers in 8051 family. **(Mar.2016)**
6) What are other microcontrollers available in 8051 family. Explain it. **(Mar.2010)**
7) What is Microcontroller? State three features of 8052 over 8051 microcontroller. **(Mar. 2004)**
7) List any six major features of 8051 microcontroller. **(Mar.2004,07,Oct.2003)**
8) Explain in how 8051 microcontroller addresses two separate memory spaces.**(Mar.05)**
9) Write any two features of microcontrollers: i) 8084 ii) 8052 iii) 8031 iv) 8050 **(Mar.2017)**

Answers Q.(1)

(a) 111 (b)4KB (c)8052 (d)8-bit(e)8051(f) 4kbyte of internal RAM (g) 8 (h) 8051
(i)8051 (j) 8051

❑❑❑

5

NETWORKING TECHNOLOGY

INTRODUCTION

During the twentieth century the most demanding technology has been information collecting, processing and distribution. Organizations with hundreds of offices spread over a wide geographical area are to be interconnected with computers for efficient gathering, processing and distribute information. This cannot be achieved by old idea of single computer. Servicing all the organization's needs is rapidly being replaced by one in which a large number of separate, but interconnected computers do this job by a system. This system is known as computer network and physical interconnecting technology is called networking technology. This topic explains the related terminologies used in networking and various systems of networking. A special attention is given to various communication media and devices used in networking.

5.1 COMPUTER NETWORKING

In simple words we can define a computer network such as "it is an interconnected collection of autonomous computers or system of computers capable of sharing resources and controlling services." The network must be efficient, reliable and economic. The common resources are to be shared by the system whenever needed and these resources include data, peripheral devices like hard disk storage, printer, fax, etc. A very common example of such a network is LAN when you are working in your college laboratory all PC's are interconnected by LAN. For big and widest organization there are other networking technologies are called MAN (Metropolitan Area Network) and WAN (Wide Area Network) respectively.

5.2 WHY NETWORK?

Many organizations use separate PCs in operation, often located apart. Each computer uses separate printer, separate program software and other essential resources. For example, a company has many departments and two or more sister organizations they have computer at each department. They may have similar software and common data files. This old system results into inefficient use of machines and becomes isolated system of machine also undergoes into lack of communication and becomes expensive due to independent resources.

At some point management has to decide to make a system which will provide communication, exchange of information access to any file and sharing software so that the system will work efficiently, reliably and economical. All these requirements can be achieved by means of networking only.

The main goals of networking are

i) Network provides resource sharing.

ii) It provides high reliability by using other machine if one machine fails in the network like military, banking, air, and traffic control.

iii) Access to any file and data.

iv) Finally the system is saving money by network only.

If we think of Wide Area Network it also provides access to remote programs, access to remote database and communication facility. For example, if a company has produced a model or product they can put all necessary information on the network so that clients can log-in over the network to see the product rates, specifications. All these applications use networking for economic reasons because the telephone call rate is expensive for long duration call but via network the line is used during the data transmission only.

5.3 NETWORKS IN COMMUNICATIONS

Now let us discuss about communication links that can be established for many to many communications a link between many PCs connected in a laboratory, telephone PBX, cable network etc. To provide this interconnection for many to many, certain technique is used known as Network.

A Network is an interconnected system that provides communication links among the two or more stations. Each station in a network is known as node. Fig. (5.1) shows a simple communications network with four nodes A, B, C and D each node is connected with other three.

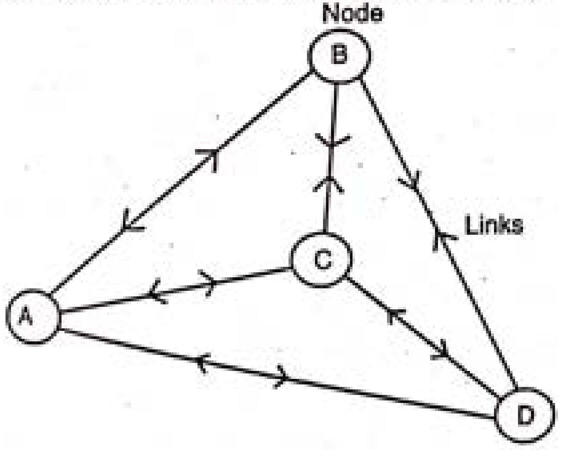


Fig.(5.1)

There are different electronic communication networks, which can be broadly classified according to their coverage area, and their facilities provision, these are:

(a) WAN (Wide Area Network)

(b) MAN (Metropolitan Area Network)

(c) LAN (Local Area Network)

As we know that today there are very common networks which are very popular and we are much familiar with those systems but we can understand their techniques by just comparing one with other as explained below

(a) WAN (Wide Area Network)

As name itself implies the coverage area is very wide, stations are very high in number hence named as WAN - Wide Area Network. A very common example is national telephone system, cellular mobile phones, pager etc. WAN is a wide network, which is capable of covering the whole country or state for communications.

In WAN, the links may be established by using cables, microwave towers or may be satellite link or it can be combination of two or many links. For example, a telephone system uses all the three media cables, microwave and satellite links for the communications. The railway reservation system, Airline reservation system these are also computer networks treated as WAN.

(b) MAN (Metropolitan Area Network)

The network smaller than WAN and bigger than LAN is also established hence known as MAN it is a medium size network. An example of MAN is a cable network within a city. Cable network is receiving TV programs from the satellite and it connects many (user). TV sets in the network to watch these programs by such a network containing so many nodes located at different locations in the network. Of course, the communication link is through a cable and one way but it is a network known as Metropolitan Area Network. Another common communication system is pager system; it is a network like MAN.

(c) LAN (Local Area Network)

The third and smallest network is Local Area Network widely used in Computer communications. When a number of computers close to each other within 10 Km radiuses are to be interconnected. LAN is the common network suitable for efficient computer communications. LAN contains 10 to 1000 PCs interconnected in an organisation with common control and storage. A computer laboratory in a college mostly uses LAN for training the students. A LAN system consists of many PC terminals but a common control terminal is controlling this network known as Server. The Server can be a minicomputer or a mainframe.

5.4 NETWORK TOPOLOGY

The network containing many nodes are physically interconnected in a certain configuration to provide an efficient communication, these configurations of interconnecting the nodes in a network is known as network topologies. While selecting a network topology the following points must be considered.

a) The cost of physical interconnections.
b) The time delay during the communications.
c) Reliability and possibilities of failure.
d) Network controlling strategy or protocols.

There are three important topologies commonly used while interconnecting computers in LAN

a) Star Network topology
b) Ring Network topology
c) Bus Network topology

a) Star Topology

This topology of network is much popular in LAN network. It is a system containing one common node as a control and many stations connected with the control node in a network like figure of star as illustrated in fig.

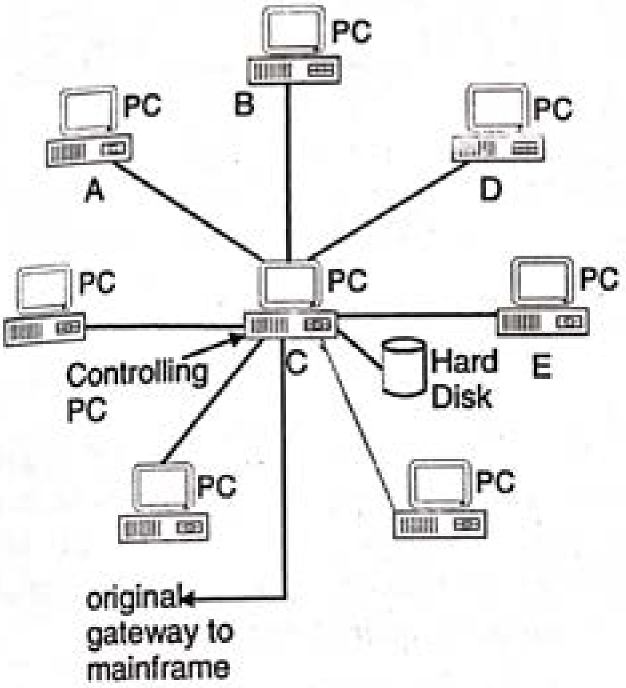


Fig. (5.2) Star topology

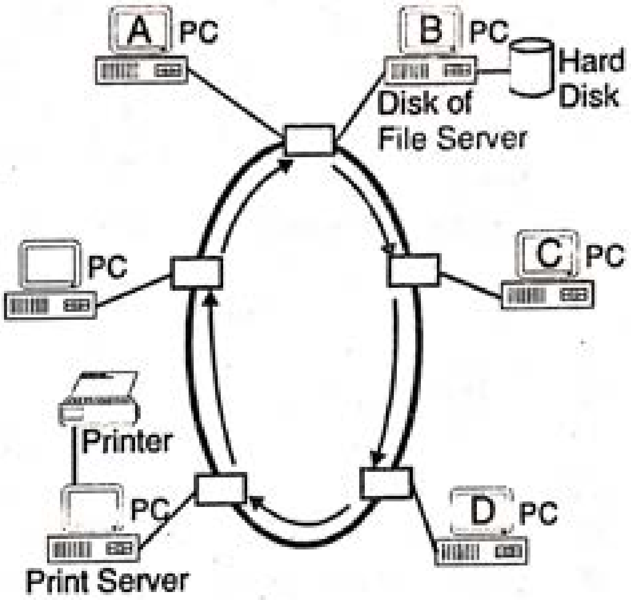


Fig. (5.3) Ring Topology

A LAN in star topology has a server or controller (minicomputer) centrally placed (Not physically) controlling different PC stations. In star topology, when a communication is to be established between A and E it has to communicate through 'C' only. The main advantage of star topology is addition of few more nodes is possible and it does not affect the time delay. But when the control node fails the whole network fails. Another common example of star is a PBX telephone system.

b) Ring Topology

The ring topology is shown in fig (5.3) where server is not centrally located. The network is formed by a number of stations with server connected one after the other forming a ring route. Each node receives data from the ring in sequence; the data is addressed in order to get the data to the desired node. In ring topology, each node is using the common ring to transmit or receive data. Suppose node A has to transmit to the node D then it is passed from A to B, B to C and then C to D the node B and C will check the address if the address is not for them they retransmit data to the next neighboring station.

The main advantage of ring topology is that each node has direct communication capacity and it is independent on one control node. But when one terminal fails the whole system fails. On the other hand its total delay in communication depends on the number of nodes and if the nodes are many in the ring more delay is introduced while communicating the data from one station to other station. These are the drawbacks of ring topology but more reliable than star topology.

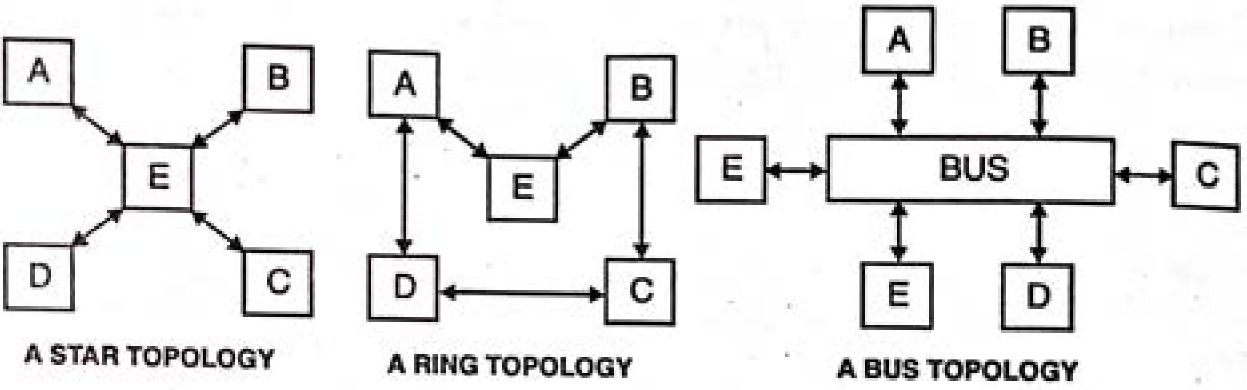


Fig. (5.4) Simplified Diagrams of Topologies

c) Bus Topology

This topology is more efficient and reliable than the star and ring topology. This topology is also known as a multi-point or multi-drop topology because in this interconnection method a common bus is used for data transmission with bi-directional communication provided by the bus for each node. The bus is available for each node to send its data to each and every computer node if desired.

Advantages

1) The bus system is much faster than other methods.
2) Direct communication between the two stations without any control station.
3) The bus topology can be extended with sub branches to form another topology known as tree topology.
4) Break down of any failure node does not affect other node's communication.
5) Bus topology widely used in wide LAN network.

5.5 PHYSICAL TRANSMISSION MEDIA CHARACTERISTICS

Physical transmission media are the physical lines or channels through which information (a stream of bits) is transmitted between computers in a network.

Before we discuss each type of transmission medium let us define their characteristics and try to understand the significance of each characteristics because each media is suited to certain types of installation than other. To choose the best type of network media for a network the following factors must be considered:

i) Cost of Media

The cost of the media must be considered first while designing a network. The cost- performance properties are the two major factors. It is decided by the user as per application and standard of the resources.

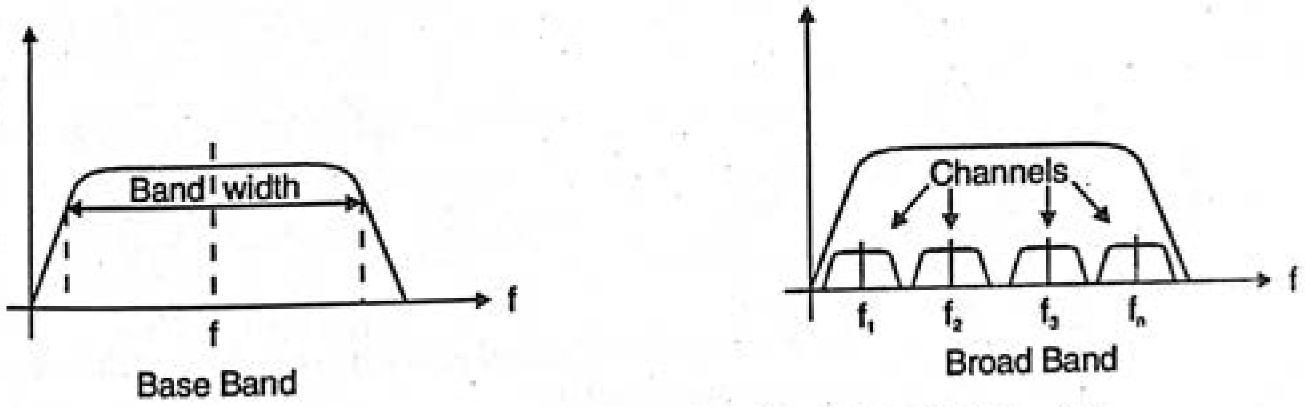
ii) Implementation Cost

When a media is to be designed for communication purpose one has to estimate the cost of cable or media as well as the cost of installation. For almost all media, the cost of installation exceeds than the cost of the cable itself. For example, for fiber optic cable special tools are to be used and by only trained technical persons can operate these tools. In other media like twisted pair the cost of installation is inexpensive.

iii) Channel Bandwidth

This is more important when media has to carry signals. It is defined as " the range of transmission frequencies that are carried effectively on a media". A very common example of bandwidth is a telephone line. Its media should have bandwidth of 3 to 4 KHz because the range of voice signals is 300 Hz-3KHz. In computer network the bit transmission rate depends on bandwidth. When digital or binary data is to be transmitted it is expressed in terms of baud rate it is measured in bits/ sec or "bps." In networking bandwidth is measured in bps.

For analog signal communication like audio/TV signals bandwidth specifies the capacity of carrying varying signals without loss of signal. The larger the bandwidth of a channel the higher is the capacity to carry information. The media can carry many signals at a time in that situation bandwidth will decide how many signals that it can carry. These concepts of carrying signals are known as base band and broad band transmission.



Base Band

Broad Band

(a) Fig. (5.5) Idea of Base band and Broadband (b)

iv) Band Usage

A channel or bandwidth is an expensive resource. In computer network except for short communication lines, many computers use the bandwidth. The bandwidth is shared so that maximum usage is obtained. The method of dividing bandwidth refer fig. (5.5) into many small channels to transmit a number of signals independently is known as multiplexing and such a bandwidth is known as broad band. On the other hand when it is for only one channel signals like telephone on a twisted wire refer fig. (5.5a) known as base band. More efficient method is broadband transmission by using separate carrier to make it distinct for each channel. Refer fig. (5.5b) f1, f2 are separate carriers allocated for each channel. The signal is modulated and not direct signal.

Comparison

Baseband	Broadband
1) Only one channel is transmitted on available bandwidth.	1) Many channels are transmitted on a available bandwidth.
2) Original information is directly sent on line.	2) Information is modulated & indirectly sent in different form.
3) Small band width can be used	3)Big band width is required to transmit more channels.
4) e.g. LAN or a telephone line.	4) Cable TV channel transmission.
5) Twisted wire cable is used	5) Co-axial cable carries braod band transmission.

v) Electromagnetic Interference (EMI)

It affects the signal which is transmitted through a media. EMI is caused by outside electromagnetic waves and also unwanted noise signals produced by various electrical appliances. EMI is interfering the signals and makes difficult for computers to decode the signal.

vi) Attenuation

Electromagnetic signals tend to weaken during transmission known as attenuation. As signals pass through the medium part of the signal is absorbed and makes the signal weak.

MULTIPLEXING

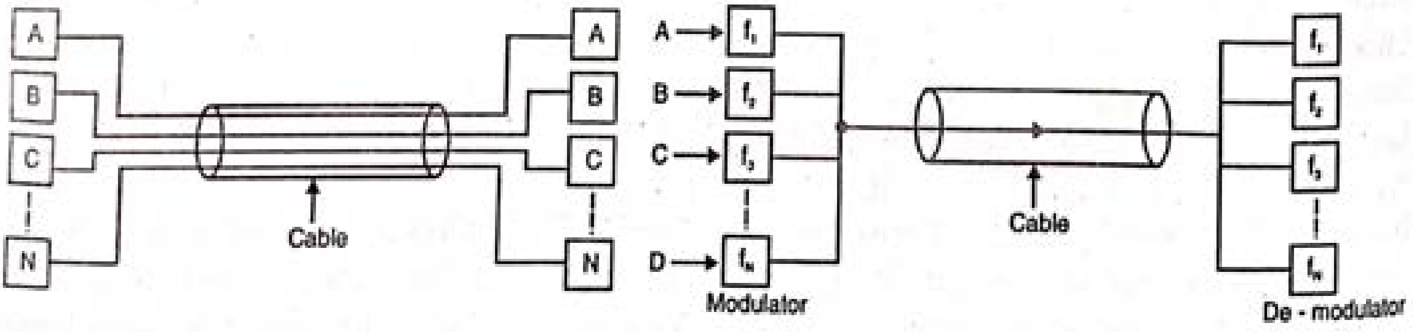
This is the technique of utilizing an available channel effectively. These are three methods of multiplexing. It belongs to broadband communication.

a) Space Division Multiplexing (SDM)
b) Frequency Division Multiplexing (FDM)
c) Time Division Multiplying (TDM)

By means of multiplexing the available channel is divided into many channels to maximize the utilization of the channel capacity.

a) Space Division Multiplexing (SDM)

This is the simplest method of physical multiplexing in which physically individual communication lines are packaged at the source.At the receiving end each line slot is separated to connect each line to individual destination as shown.



(a) Fig. (5.6) Idea of SDM and FDM (b)

The advantage of this method is that it allows individual line to each user & it uses base-band for short distances. Another advantage of SDM is that the system is simple to install and use.

b) Frequency Division Multiplexing (FDM)

This is the method of multiplexing where the medium is carrying multiple individual channels instead of using separate Physical wire for each channel. It means that in FDM physical channel carries a number of small logical channels on a common cable. In fact each channel is transmitted by means of modulation & at the receiving end it is demodulated for this purpose it requires modem. In FDM each channel sends information by using a separate carrier frequency as shown in the figure f1, f2, …… fn.

Since each carrier frequency is different, they run simultaneously through the cable. At the receiving end they are separated by means of demodulator.

c) Time Division Multiplexing (TDM)

TDM is a very popular modern technique of utilizing the capacity of a physical channel effectively. As name suggests in TDM each user of the channel is allowed to transmit for a small time interval after every periodic time. So that user can utilize full bandwidth in allocated time slice. As shown in the figure the physical channel carries information sequentially such as A3, A2, A1 and received in sequence as A1, A2, A3.

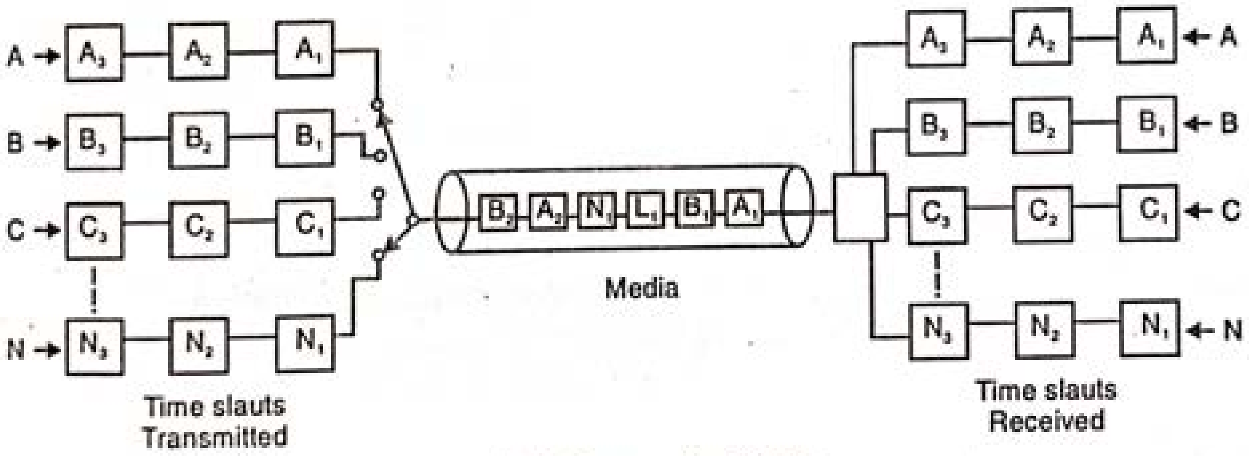


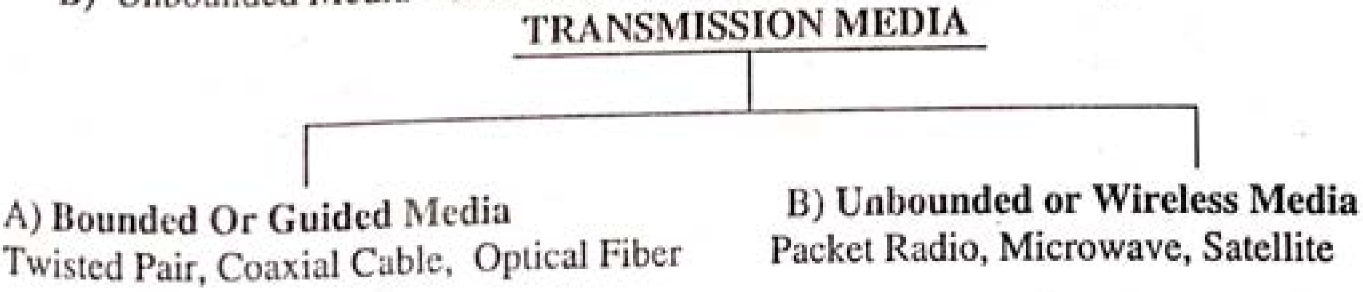
Fig. (5.7) Concept of TDM

5.6 STUDY OF TRANSMISSION MEDIA

Media can be broadly classified in to two groups.

A) Bounded Media – like twisted wires, Co-axial cables, fiber optic cables.

B) Unbounded Media – like radio waves, microwave, infrared etc.,



TRANSMISSION MEDIA

A) Bounded Or Guided Media
Twisted Pair, Coaxial Cable, Optical Fiber

B) Unbounded or Wireless Media
Packet Radio, Microwave, Satellite

3. Fiber - Optic Communications

The conventional electrical wave communications as discussed in this topic was with the help of copper cables and by radio communication through air or space using a high frequency carrier. These communication systems facing the two important problems till today;

i) Limited information carrying capacity due to limited bandwidth.
ii) External noise interference resulting in poor quality communications.

Recently, in late nineteenth century one scientist proved that using light waves instead of electrical waves for carrying information could easily solve these two difficulties. The light wave can be efficiently conducted through transparent glass filer cables known as optical fiber cables. The communications carried out by optical fiber cable and using light wave is known as fiber optic communications and it is becoming more popular in telephone communications as well as other digital communications systems of networks.

Advantages over conventional cables

1. Fiber optic cables can carry much information at a time. It provides widest bandwidth. The data rate is 100 Mbps to 2 Gigabits/sec.
2. Fiber optic cables are very small in size as shown in fig.(5.9) so that many cables can be crowded to carry more and more information in a small space.
3. Fiber optic communication uses light source of extremely high frequency $3x10^6$ GHz and with a very high velocity it is the fastest communication with maximum bandwidth.
4. In fiber optics, the transmitters and receivers are very simple and provide reliable communications.
5. With fiber optic cables the signal can be transmitted over long distances with negligible attenuation.
6. These are lightweight cables having very small diameters than conventional copper cables.
7. Since they are prepared by fiber material like silica glass, silicon di-oxide they are electrically bad conductors they are free from electrical interference resulting in noiseless communications and also provide shockproof communications.
8. Fiber optic cables cannot be tapped like cable TV signals and also they do not radiate signals they provide secured communications.
9. As compare to copper cable, fiber optic cables are very strong, flexible, can stand with high temperature and do not show corrosion due to water or chemicals.
10. Fiber optic cables can be used effectively for variety of communications systems.

The System Of Fiber Optic Communications

Fiber optic communication is a light wave communication. A light signal is passed through a fiber optic cable known as light pipe or light guide. The light signal propagates through fiber cable by multiple reflections in zigzag path. Some part of the signal is lost due to internal absorption. Light is a signal it is an electromagnetic wave like radio wave. The light pipe carries many signals like many telephone channels through the cable without any interference.

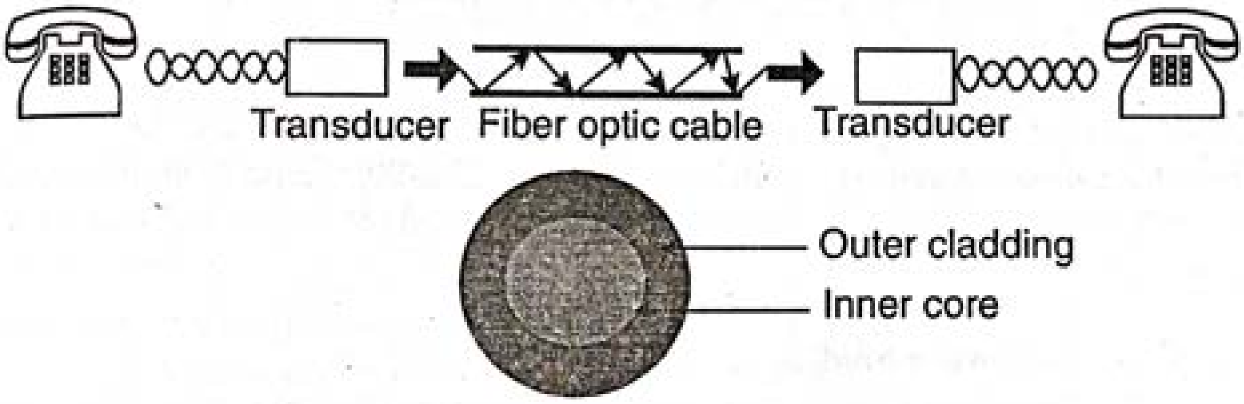


Fig. (5.10) Fiber optic communications system

Comparison Of Bounded Cables

Properties	Twisted pair	Co-axial cable	Fiber optic cable
a) Cost	Inexpensive	Twice or thrice than twisted pair	Expensive
b) Installation	Easy	Easy	Difficult
c) Attenuation	More	More	Very less
d) EMI effect	Maximum	Minimum	No effect
e) Band width	1 to 100 mbps/100 mtr.	500 mbps/ 100 mtr.	Gega bps/km.
f) signal type	Electrical	Electrical	Light signals.

B) UNBOUNDED (WIRELESS) MEDIA

In this group of wireless media radio waves in the very high frequency (VHF) band which are not used for any other communication may be used for communication between computers.

Advantages:

1) The advantage of wireless media is high data rates by using large bandwidth which can give transmission speed around 24 Kbps.
2) By this media the communication can reach rural and hilly areas.
3) Bandwidth for digital data 1 to 10 Mbps.

Disadvantages:

1) VHF waves are corrupted by atmospheric noise and error may introduce in data and affected by EMI.
2) Lack of security, because radio waves may be received by any one.
3) The installation cost is very high and installation is complex.
4) Attenuation is very high it can be minimised by using repeaters.

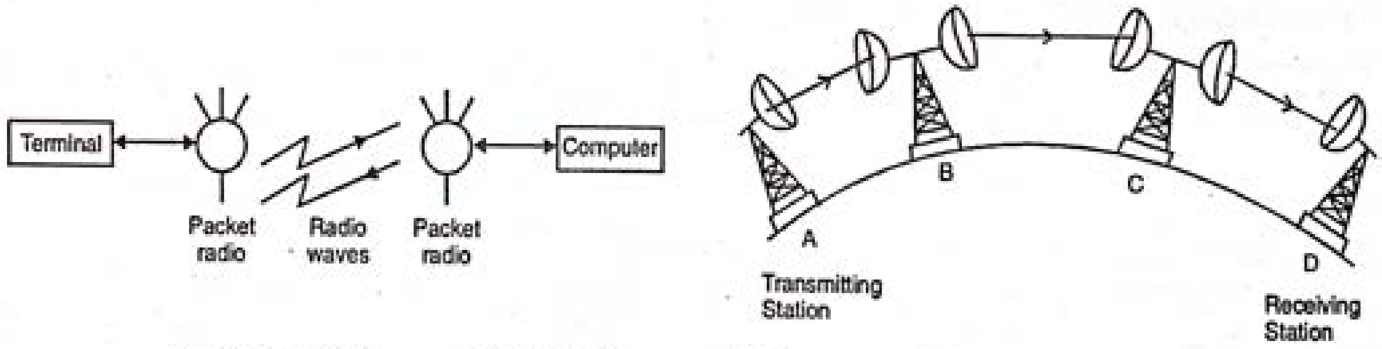
There are three types of unbounded media communication:

1) Packet radio

It is a combined packet radio having both transmitter and receiver with different frequencies. A packet radio is attached to each terminal and the computer. The information entered on a terminal is transmitted using the packet radio and received by the computer. Information is processed and results are transmitted back to the terminal by the computer.

2) Microwave Media

Microwave communication use wave guides and repeaters. Microwave is extremely high frequency about 100 Ghz. Since microwave can also bend or pass obstacles like hills. For this purpose repeaters are used between two hills they are placed in a line of sight.



Packet Radio **Fig (5.11)** **Microwave Communications**

3)Communication Satellite Media

Microwave signals like television; the ionosphere cannot reflect radar signals. Only way to transmit these TV signals to transmit the signals directly from the tower to the receivers. But due to direct transmission of such a high frequency signals they can propagate up to only limited area only in the line of sight area. It happens due to earth's curvature that's why TV towers are located on top of the hills. As illustrated in fig (5.11) the area can be increased by making multi- trans-reception where signals can be sent from place A to D by making transmission from tower A to B then B to C and C to D, in this process the cost and signal strength problems become main difficulties. The tower B and C are called repeaters. Even if to transmit these TV signals from one city to another city located about 200Km it becomes essential to place many repeaters if the area is having number of hills and mountains in between the two cities.

To overcome these difficulties one master repeater can be launched in to the space by making it geostationary. It is a satellite, which can be sent in to the space so that it can cover maximum area of the earth's surface as shown in fig (5.10)
In this way a communication satellite is an artificial satellite orbiting round the earth in the same direction as the direction of the earth movement and with the same velocity to make it stationary. In other words **"a satellite is an electronic communication system that orbits around the earth"**. Now the idea is very clear that the TV signals are sent from the tower on the earth to the satellite in the space. The satellite amplifies these signals and re-transmits them back to earth. In satellite communication, the signal communication is in full duplex the earth station and satellite both contains transmitter and receiver.

About the satellite

- A satellite is an artificial revolving object round the earth.
- It contains different communication electronic systems
- It is launched for long distance as well as microwave communications
- The satellite is a full duplex system since it is a two-way communication
- Satellite communications increase the area of coverage
- It can be used for a variety of useful applications like telecommunication, meteorological information, cellular phone etc.

Fig (5.12) shows the basic use of communication satellite where the stations on earth are known as earth stations or ground stations and the satellite as a remote station since it is in remote area as space.

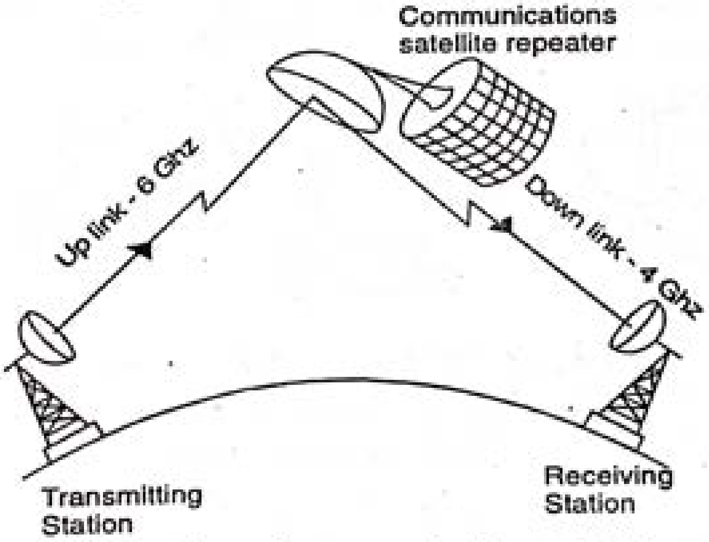


Fig (5.12) Use of communication satellite

The operating steps are:

1. An earth station transmits information to the satellite by using a carrier known as "up link" frequency.
2. The satellite receives the up-link it amplifies it.

3. The satellite transmits the amplified information signals by re-transmitting it on another carrier frequency called as "down link" frequency towards another earth station on the earth.

5.7 NETWORK CONNECTIVITY

A small network normally grows but it can not grow beyond certain limit it affects its performance like drop in printing speed, machine's response etc. It needs network expansion. Typically two main types of expansion are needed

- Expansion within a network called network connectivity
- Expansion of two or more networks called internetwork connectivity

To expand a single network without breaking in into new parts or connecting it to other networks some standard connectivity devices are used such as:

1. Hubs
2. Modem
3. Repeaters
4. Routers

1) HUBS

This is the simplest connectivity device used to extend a network. All networks (except those using co-axial cable) requires a central junction are called hubs.

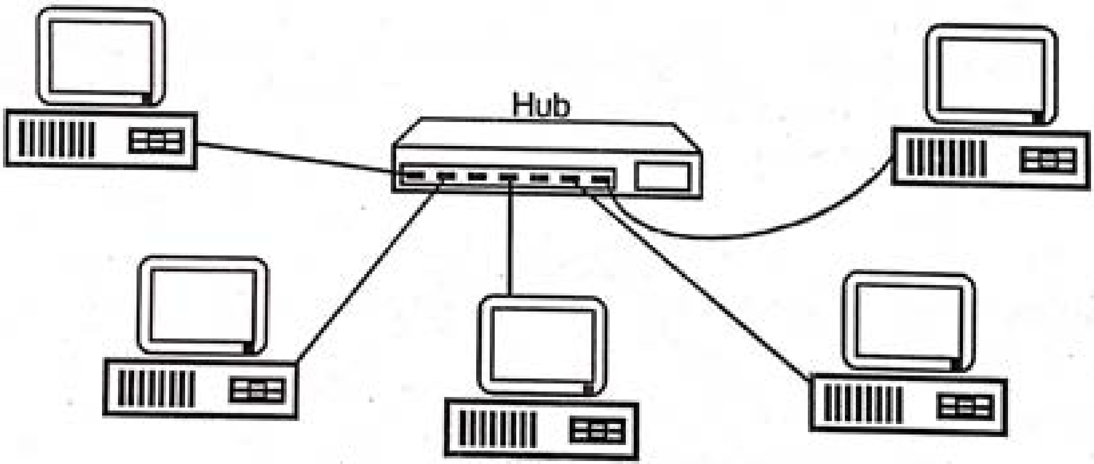


Fig.(5.13) Use of Hub in a Network

Fig.(5.13) shows a simple network inter-connected with hub. Hub is a structure with which cables can be connected without soldering wires. There are three types of hubs:

a) Passive hub: It simply combines the signals of network lines. There is no amplification or boosting of signals.

b) Active hub: Since this device not only inter-connects but also amplify and clean-ups signals, it is known as active hub. This hub contains electronics circuits.

c) Intelligent hub: In addition to signal amplification and regeneration, this type of hub performs intelligent work like network management and intelligent path selection.

2) MODEMS

If you are using computers for Internet or E-mail communications then you must have heard about Modems. Modem is a technique of using a telephone line for digital communications. If you want to use computers for communications may be for short distances or very long distance communications the Modem is the technique that is used to interconnect computers by using a very common media telephone line. Suppose you want to use radio communication technique then every one has to use receiver and transmitter which is impossible. Secondly if you use a telephone line directly for transmission of digital data the line is not compatible due to its ability of transmitting audio tone or analog signals with very small bandwidth of about 3 KHz. The following main problems involve while directly using a telephone line

i) The telephone line produces electrical disturbance.
ii) Small telephone line -bandwidth.
iii) Capacity to transmit only analog signals like audio signals. (300 Hz to 3 KHz)
iv) The analog circuitry is not compatible for digital data transmission.

To overcome these difficulties a new technique is employed *modem* it stands for modulation - demodulation techniques. The electronic equipment, which provides this technique, is known as modem. When digital information is to be transmitted over long distances by using a telephone line it becomes necessary to convert digital data into analog signal (within voice frequency range) before transmission on a telephone line by using modulator. At the receiving end it is necessary to convert information from analog to digital by means of demodulation. While using the telephone line for digital communication in full duplex or two way communications modulator - demodulator unit is used at both ends. Modem is necessary at both computer stations.

Modem Process:

Let us see how modem is functioning for digital communication between the two computer terminals located at two distinct (long distance) places. Refer Fig (5.14) to understand its simple function

Digital communication between the two computers A and B is carried out in the following steps

1) The user at computer 'A' sends data in digital form. The computer 'A' is modulating digital data into analog form; the modulation technique may be FSK (Frequency Shift Keying) type.
2) Modulated signal is within telephone frequency range it is transmitted over telephone line by selecting a proper dial code.

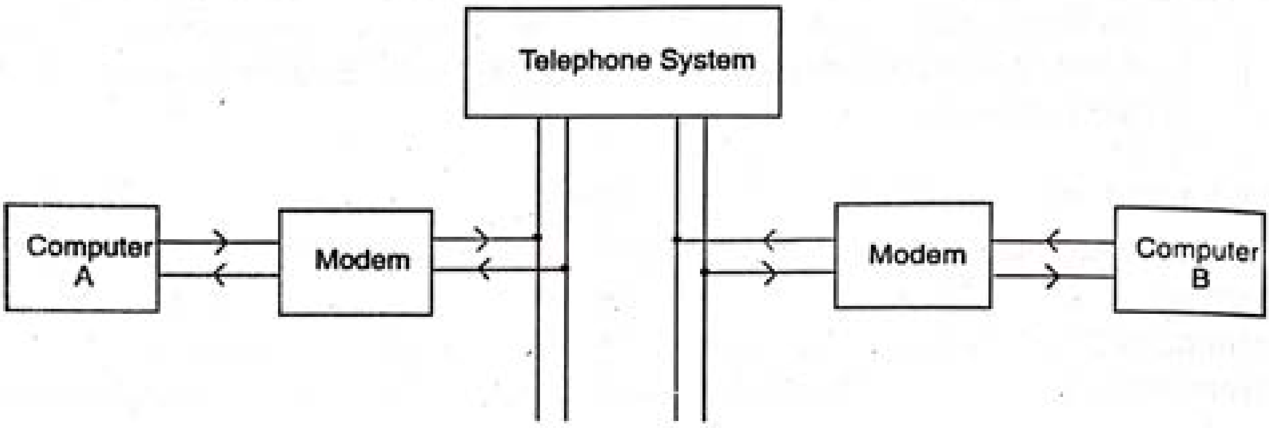


Fig. (5.14) Use of Modem

3) The computer ' B' terminal is selected by its dial code.
4) The modem of computer 'B' is converting analog modulated signal into original digital data by the process of demodulation.
5) The demodulated digital information is processed and displayed on the screen of terminal B.

An opposite and similar action is carried out while transmitting the data from Computer 'B' to Computer 'A'. A simultaneous communication is possible by using two separate frequencies to avoid interference and to provide full duplex communications.

Applications of MODEM

Modems are specially designed for the following modem communication system; it is becoming more useful for

1) Fax communications	2)E-mail Communications
3)Chat Communications	4)Internet browsing

3) REPEATERS

As name itself suggests it expands network length. If the network is to be used beyond pre-defined limit repeater is used They simply regenerate physical signals on a network as shown in the figure. Repeaters can be of both types. The first type is amplifying the entire signal, while the second type where repeaters, creates an exact duplicate of signal known as signal regenerating repeaters.

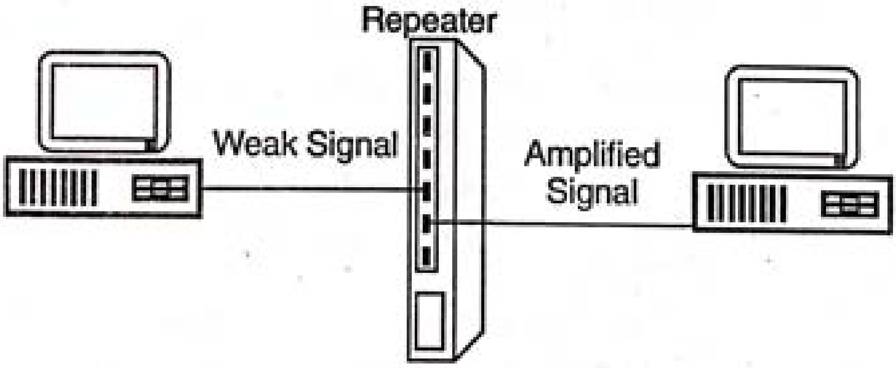


Fig. (5.15) Role of Repeater

4) ROUTERS

This is the inter network connectivity device used to inter-connect two or more independent networks. Router is a combination of hardware and software. The hardware can be a network server, a separate computer or a special black box device. The software like algorithms are used to determine the best path by which to send a signal packet. It also include operating system.

Figure shows the role of repeaters in an inter network consisting of four networks. Routers use logical and physical addressing to connect two or more logically separate networks. Router are used:

a) to divide a big network into small networks called subnets.
b) to connect a small network to big network like LAN to WAN.

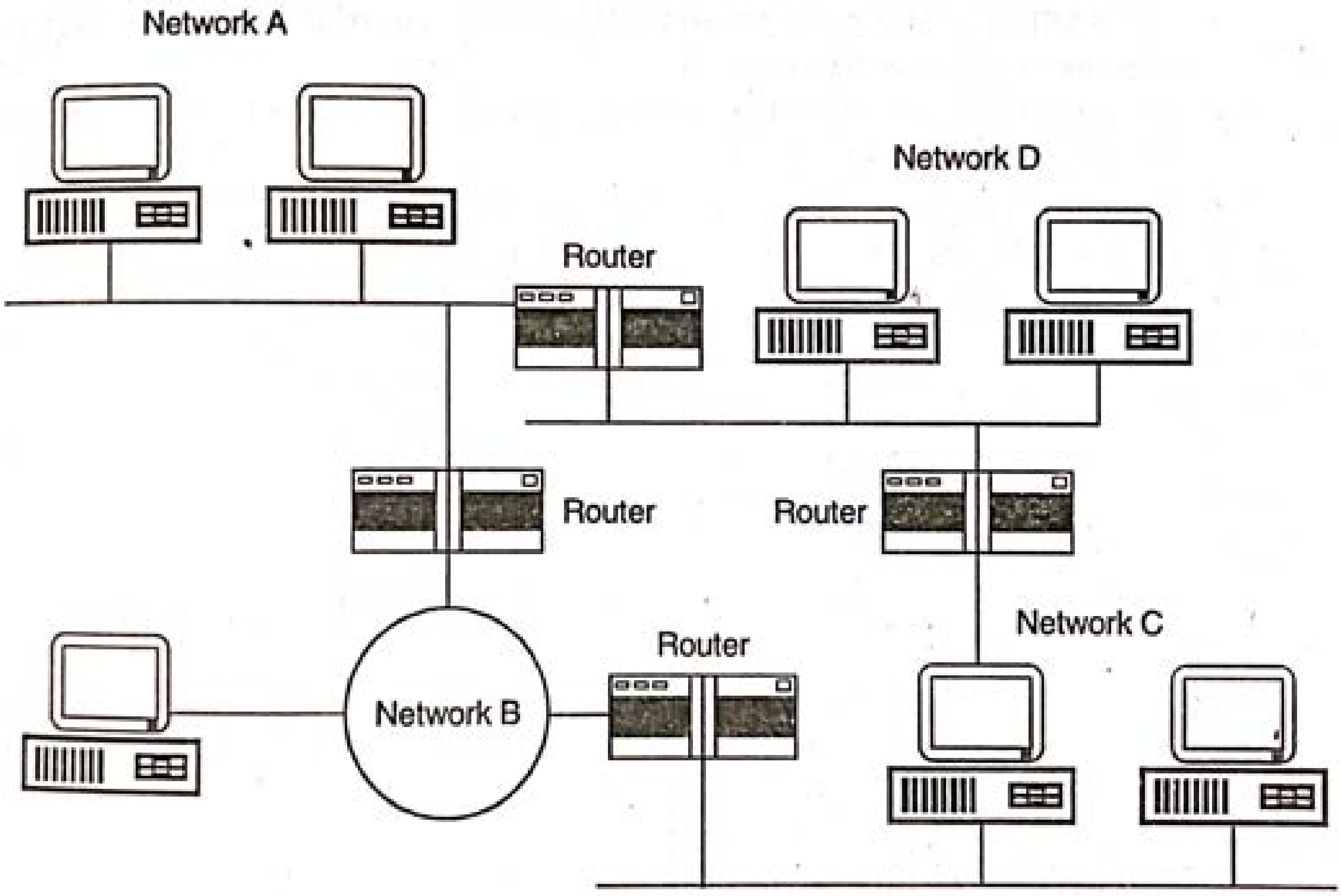


Fig. (5.16) Role of Router

A router maintains a table of available routers and their status. Router use this information along with routing algorithm to determine the best route.

The routers can be static type or dynamic type. The static routers do not route path themselves while dynamic routers determine path themselves.

5.8 NETWORK ARCHITECTURES

When you work on networks you will come across several network architectures. These architectures are designed to solve specific problems and each has physical layout of connecting devices, cables, network cards and protocols. The two popular architectures are discussed here:

1) Ethernet

Ethernet is a most common physical network architecture in use today. Ethernet is a simple method of connecting many computers together in a LAN. It is a bus or star bus topology using baseband signaling. Ethernet use passive medium like twisted pair, co-axial cable. According to their data transmission speed and physical media type there are standard types of Ethernet such as:

- 10 BASE 5: Speed 10 Mbps(10) and thick co-axial known as thicknet. Maximum cable net 500 mtrs.(5).
- 10 BASE 2: Speed 10 Mbps(10) and thin co-axial cable known as thinnet. Maximum length 200 mtrs.(2)
- 10 BASE T: Speed 10 Mbps(10) and uses UTP twisted pair 100 mtrs.
- 10 BASE FL: Speed 10 Mbps(10) uses (FL)
- 100 BASE VG: Speed 100 Mbps(10) twisted pair.(VG-Voice Grade)

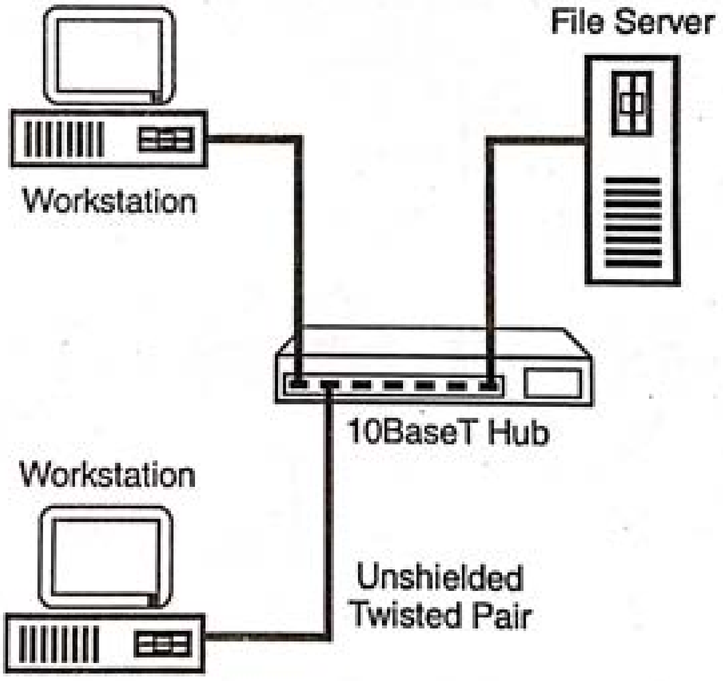


Fig. (5.17) Ethernet Architecture 10BASE T HUB

Role of Ethernet

In case of Ethernet workstations send signals in the form of packets across the network. When a collision takes place it stops transmissions of workstations and allows to re-transmit after random internet of time. Ethernet uses CSMA/CD protocol it is Carrier Sense Multiple Access with Collision Detection Protocol.

2) Token ring

Token ring architecture was developed by IBM as a high reliable network. It is more complex than Ethernet. It uses physically star topology but logically ring topology. It uses physically UTP or STP cables and also can use fiber optic cable to extend a network.There are different types of token ring cabling as per IBM standards.

Type 1: Uses STP cables and used within the same building.

Type 2: It uses twisted pairs and allows to use it for both telephone voice signals and computer data. In the same physical area or room.

Type 3: Uses UTP or STP but for limited distance.

Type 5: It uses fiber optic cable but only on the main ring path.

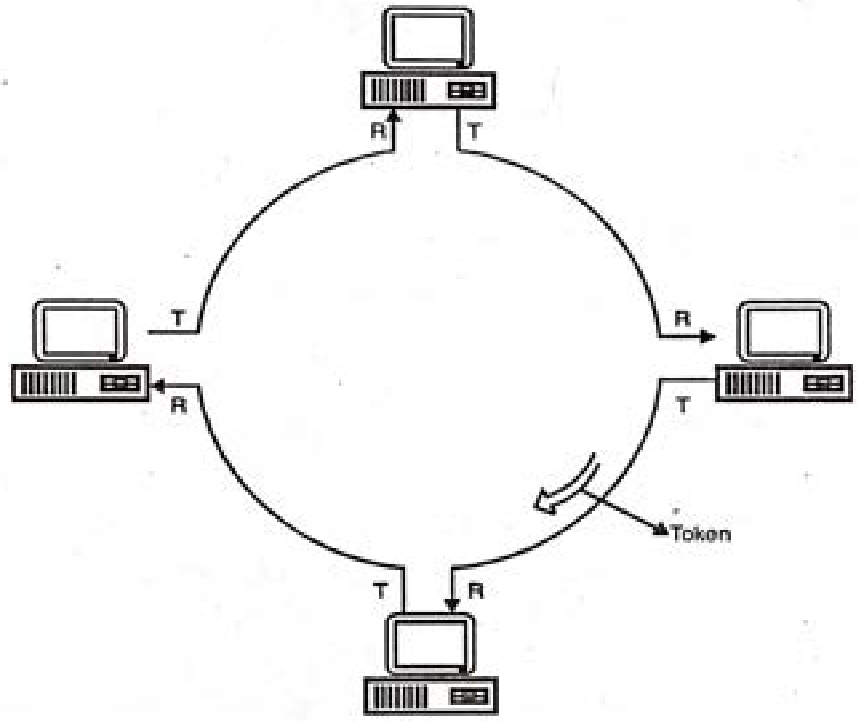


Fig. (5.18) Token ring Cabling

Role of token ring

In this architecture each workstation is attached to a controlled card and all workstations function logically in a ring. Each in the ring acts basically as a repeater. The ring passes a free token (a small frame with a special format) around the ring in one direction. A station in the ring receives the token from its nearest upstream neighbor, when a station receives a free token, it knows it can attach data and send it to nearest down stream neighbor in the ring. If the station has no data it will pass receive data. In token ring, each station is given an equal chance to have the token and take control in order to pass data.

5.9 PROTOCOLS

These are the set of rules or standard designs to enables computers to be connected to each other and transfer information through a network. When the network is too big for the communication, a well understood standard method of communication and physical inter connection should be established. When computers in different countries are to be connected together then their must be different standards of different nations. It becomes very complex communication. This problem can be solved by making common rules to inter connect and communicate between computers are known as **protocols,** all the entities in the network must agree on these protocols.

In such a network data is transmitted in the form of packets on the network. The packet contains standard format. The format is decided by the type of protocols. The format of packet in a type of protocol specifies the following information such as:

Header	Data	Trailer

a) **Header: (Start of packet)**

It specifies source and destination address.

b) **Message: (Data)**

This part of packet contains actual data or information. The information coding also decided by protocol standard. e.g. data, nibbles or in bytes or in ASCII or EBCDIC and so on.

c) **Trailer:**

In this part of packet, ending information is transmitted like error checking and correction information.

When this packet is transmitted during the transmission, message may get damaged or loosed due to serious reasons, for this purpose this information is sent in the trailer part.

There are different international standard protocols at different levels.

a) TCP/IP (Transmission control Protocol and Internet Protocol), etc.
b) UDP (User Data Protocols)
c) ARP (Address Resolution Protocols)
d) ICMP (Internet Control Message Protocols)

Internet Protocol:

The Internet protocol was developed by United States Department Of Defense to interconnect educational institute and government installations. The protocol, which has become extremely popular, is known as TCP/IP (Transmission control Protocol and Internet Protocol). This Internet protocol do not belong any one company and the technology is available to everybody. Due to this Internet protocol is supported by the widest variety of vendors.

TCP/IP Protocols:

TCP/IP Internet protocol use three types of addresses for network addressing

1) Hardware or physical address, it is used by the data link and physical layers. It is hard coded into the network cards at each device.
2) Internet protocol address provides logical node identification. This address is unique address assigned by an administrator according to certain guidelines. It is expressed in four parts dotted - notation.

e.g. 123 · 144 · 131 · 12

3) Logical node names, which an administrator can assign, are easier to remember than an IP address.
   e.g. BARNEY.COM

IP Internet Protocol

IP works at the network level. It is a connection less data from protocol. IP uses packet switching and performs route selection by using dynamic routing tables that are referenced at each hop. This packet making up a message could be routed differently through the Internet work depending on the state of the network at each hop. e.g. If a link were to go down or become congested, packets could take a different route. Internet Protocol (IP) performs the following functions and uses the methods as follows:

1) For addressing, IP uses the logical network address.
2) For switching purpose, it uses the packet switching method.
3) For route selection, it uses the dynamic method.
4) For connection services IP provides error control.

ACCESS METHODS

As explained earlier when more than one node try to use transmission media it results in collision. To minimise or avoid collisions there are certain methods called access methods. There are three techniques as follows:

i) Contention ii) Polling iii) Token passing

i) Contention:

Contention means to struggle; nodes are struggling for accessing media. This is more useful in LANS. In contention-based network each node is observing media before transmitting. There are two methods known as **carrier sensing** and **carrier detecting.** In carrier sensing method, each node listens or sensing the media if media is busy it is waiting till it becomes free and then starts transmitting. On the other hand in carrier detecting method, nodes are continued to listen media and if another carrier is detected then it stops and waits for random amount of time. Applying these methods the number of collisions is reduced.

ii) Polling:

In polling based method the network traffic is controlled by a device called controller, which polls the node that is ready. It is continuously checking that which node is ready to transmit or receive and polls down to media.

iii) Token passing:

As explained in token ring architecture it is forming a ring and circulating token known as frame to each node. If node is ready at the moment when it has token it occupies media to transmit or receive. After finishing its work, it passes token to the next node in the ring.

QUESTIONS

1) Select the correct alternative and rewrite the sentence:

   1) If the network is to be extended beyond predefined cable limit _____ is used.
      i) Modem, ii) Repeater, iii) Hub, iv) Router **(Mar. 2002)**

   2) _____ cable type is ideal for connecting between two buildings.
      i) UTP, ii), STP, iii) Co-axial, iv) Flat **(Specimen Paper)**

   3) Token ring belongs to _____ Topology
      i) Star Ring, ii),Star Bus STP, iii) Mesh, iv) A Star Bus

   4) _____ cable has highest bandwidth
      i) UTP, ii), STP, iii) Co-axial, iv) Fiber optic

   5) While installing an Ethernet 10BASE T network _____ cable is required
      i) Twisted pair, ii), Thinnet, iii) Co-axial, iv) Fiber optic

   6) The transmission rate of _______ is typical for fiber optic cables. **(Oct.03)**
      i) 10 Mbps, ii) 25 Mbps, iii)100 Mbps, iv) 5000 Mbps

   7) A device used for modulation and demodulation process in network is _____.
      i) Hub, ii) Router, iii) Modem, iv) Repeater **(Mar.2004)**

   8) The transmission rate of __________ is typical for the fibre optic cable.
      a) 10 mbps b) 25 mbps c) 100 mbps d) 500 mbps **(Mar. 2005)**

   9) _________ does not regenerate the computer signal in networks. **(Mar. 2006)**
      a) Passive hub b) Active hub c) Repeater d) All the three

   10) _______________ cable has highest bandwidth. **(Mar. 2007)**
       a) UTP b) STP c) Co-axial d) Fiber optic

   11) Most widely used and economical cable for network installation is_____
       a) Fiber-optic b) UTP c) STP d) Co-axial **(Mar. 2008)**

   12) If length of cable is very long then —— is used in between the weakened signal to its original level. **(Mar. 2016)**
       a) MODEM b) HUB c) REPEATER d) ROUTER

2) What is transmission media? Explain in short six characteristics of Transmission media. **(Mar. 2017,2019)**

3) Compare the characteristics of fiber-optic cable and UTP cable. Mention at least three points. **(Mar. 2002)**

4) Define Bus, Ring and Star Topologies. Draw simple diagrams. **(Mar. 2002,2016)**

5) What is meant by protocol? Explain the concept of TCP/IP protocol. **(Mar. 2002,Oct.03, Mar.04,07,19)**

6) Explain any four points to explain why wireless networks are useful? **(Mar. 2002)**

7) Write the functions of each of the following devices in short:
a) Modem, b) router, c) hub, d) repeater (Mar. 2002,Oct.03)

8) Compare characteristics of fiber optic cable and co-axial cable. Mention at least three points. **(Specimen Paper)**

9) Define topology. Explain STAR and RING topologies with diagram. **(Specimen Paper,Mar.06)**

10) Write a short note on unshielded twisted pair cable with its characteristics. **(Specimen Paper)**

11) Explain the following characteristics of transmission media:
a) Bandwidth, b) band usage, c) attenuation **(Specimen Paper)**

12) Explain hub and repeater in detail. **(Specimen Paper)**

13) Write any four advantages of fiber optic cables

14) Why satellite communication is preferred than other wireless communication?

15) Explain WAN, MAN and LAN. (Mar. 2016)

16) Compare the Coaxial cable with Twisted Pair cable. Mention at least three points. (Oct.03,Mar.05)

17) Explain the ring topology and token passing. (Oct.03,Mar.06)

18) State four LAN wireless transmission methods. Explain any two. (Oct.03)

18) Explain in short the six important characteristics of transmission media. (Mar.04)

19) What do you mean by network topology? Explain in brief the two basic categories of topology. (Mar.04)

20) Compare any four attributes of UTP and Optical Fiber Cable. (Mar.04)

21) Write a short note on Ethernet. Mar.05,2016)

22) What is a Hub? Explain its types. (Mar.05,2020)

24) Explain router and Modem with their uses. (Mar.06)

25)What do you mean by Networking? State any three differentiation points between LAN and WAN. (Mar.06,2019)

27) Explain any four points to explain why wireless networks are useful? (Mar.07)

28) Discuss any two Access Methods of Networking. (Mar.07)

29) Write a short note on MODEM. (Mar. 2016)

30) Explain the structure of fiber optic cable. (Mar. 2016)

31) Define Topology. Explain Physical and Logical Topology. (Mar. 2020)

Answer Q. (1)

1) Repeater 2) Co-axial 3) Star Ring 4)Fiber optic 5)Twisted pair 6)100 Mbps

7)Modem 8)100 mbps 9)Passive Hub 10)Fiber optic 11)co-axial. 12) Repeater

❑❑❑

PAPER-II
PRACTICALS

HINTS FOR PRACTICALS (PAPER-II)

While working on a microprocessor kit you are required to write programs in assembly language. Write the instructions in programs in mnemonic form. This is called assembly language. Once written in assembly language codes the program in machine codes (Machine language).

Writing a program is equivalent to giving specific commands to the microprocessor in sequence, to perform a task. This process can be divided into following steps.

Step 1: Read the problem carefully
Step 2: Break it down into small steps
Step 3: Represent these small steps in possible sequence with a flowchart
Step 4: Translate each block in flowchart into appropriate mnemonic (assembly language) instructions
Step 5: Translate assembly language program to machine code (machine language). Steps1 to 4 accounts for assembly language program development.
Step 6: along with steps 1 to 4 accounts for the development of executable machine language program.

Illustration: We will now apply this approach to solve following problem.

Problem statement: Write a program in 8085 assembly language to add two 8-bit registers (sample Program).

Problem analysis: The problem can be broken into following subtasks.

Step no.	Task
1	Load register A and B with any two nos.
2	Copy data of B in another register.
3	Add contents of B with A

We now ask following questions for the step indicated.

a) For Step 1: Is there an instruction available to load register with data
b) Answer: Yes, Use MVI instruction
c) For Step 2: Is there an instruction which will copy contents of register B to D
d) Answer: Yes, Use MOV instruction
e) For Step 3: Is there an instruction which will add contents of register B to A
f) Answer: Yes, Use ADD instruction

Flowchart:

Draw the flowchart symbolizing flow of actions to be taken.

Program Writing:

The final assembly language program can be written in a standard format this format includes following columns.

a) Address b) Machine code c) Label d)Mnemonic/Operand e)Comments

Normally columns c,d,e are prepared first and after coding the program columns a and be are completed. **(Refer OPCODE Chart Page no. 307 of this book)**

(Journal Write-up)

Address	Opcode	Label	Mnemonic/ Operand	Comment
4000	3E	START	MVI A, 11H	Load register A with 11H
4001	11			
4002	06		MVI B, 12H	Load register B with 12H
4003	12			
4004	50		MOV D, B	Copy data of B to D
4005	80		ADD B	Add B to Accumulator
4005	CF		RST 1	Jump to saving Routine

After completing the program writing, connect the microprocessor Kit to power supply and enter the above program from address 4000H just by entering opcodes. Execute the program by operating routine provided with Kit.
Check the contents of Register A, B and D
You should get A =23H, B=12H and D=12H

Example: Consider an example of Memory Block Transfer
Source Address = 4300 to 43FF Destination Address = 4400 to 44FF

Important Note:

1. Refer this sample program given below while entering three byte instruction like LXIH4300H enter in the sequence (Opcode 0f LXIH=21), (Lower byte= 00) and then (Higher byte= 43)
2. While writing the Branching instructions like JUMP write the address of Labeled Loop e.g. in the given program to enter JNZ STORE enter in the sequence(Opcode 0f JNZ= C2), Address of STORE Loop (Lower byte= 08) and then (Higher byte= 40)

Program:

Address	Opcode	Label	Mnemonic/ Operand	Comment
4000	21		LXI H, 4300H	;Starting Address of
4001	00			;Source in HL
4002	43			
4003	11		LXI D, 4400H	;Starting Address of
4004	00			;Destination in DE

4005	44			
4006	0E		MVI C, 64	;Set no.of bytes in C
4007	64			
4008	7E	STORE	MOV A, M	;Load Source data
4009	12		STAX D	Transfer in Destination
400A	23		INX H	Increment Source
400B	13		INX D	Increment Destination
400C	0D		DCR C	Decrement Counter
400D	C2		JNZ STORE	Repeat until C= 0
400E	08			
400F	40			
4010	CF		RST 1	Stop

ASSEMLY LANGUAGE PROGRAMS

Expt1: ADDITION

a) Write a program that adds the contents of a block of memory using the DAD instruction. Block length is stored at 4050H and starting address of block is 4051H. Store the TWO-byte result below the end of the block. (20)

b) Enter the program on the microprocessor kit.

c) Execute the program. Write the contents of the data, memory locations before and after execution as well as the contents of the registers used in the program after execution and also the bit contents of all five flags individually. Verify the results.

Addrs	Opcode	Label	Mnemonics	Comment
4000	11 00 00		LXI D, 0000H	;initialize DE = 00
4003	21 00 00		LXI H, 0000H	;initialize HL = 00
4006	01 40 40		LXI B, 4000H	; set data pointer BC=4040H
4009	0A	UP	LDAX B	; first data in Acc.
400A	5F		MOV E,A	; store it to reg.E
400B	19		DAD D	;direct addition (HL+ DE)=(HL)
400C	03		INX B	;increment data pointer
400D	79		MOV A,C	;lower address byte to Acc.
400E	FE 45		CPI 45H	;compare with lower byte of last address
4010	C2 09 40		JNZ UP	;Jump if not finished to UP
4013	7C		MOV A, H	;move lower byte result in Acc
4014	02		STAX B	;store at next mem. Address
4015	03		INX B	;increment data pointer
4016	7C		MOV A,D	;store upper byte of result
4017	02		STAX B	;store at next mem. Address
4018	CF		RST1	;stop

Data input: Result of addition:

4040	90H	4045	**30H**	Lower byte of addition
4041	90H	4046	**03H**	Higher byte of addition
4042	90H			
4043	90H			
4044	F0H			

Expt.2: SUBTRACTION

a) Write a program that subtracts the number stored in 4021H from the number stored in 4020H. Store the absolute difference in memory location 4022H as the result. (20).

b) Enter the program on the microprocessor kit. (5)

c) Execute the program for a positive as well as negative difference. Write the contents of data memory locations before and after execution as well as the contents of the registers used in the program after execution and also the bit contents of all five flags individually. Verify the result.

Addrs	Opcode	Label	Mnemonics	Comment
4000	21 20 40		LXI H, 4020H	; Memory Pointer
4003	7E		MOV A, M	; Data in accumulator
4004	23		INX H	;Increment memory pointer
4005	46		MOV B, M	; store next data to reg.B
4006	98		SUB B	;Subtract contents of B from Accumulator.
4007	23		INX H	;Increment memory pointer
4008	77		MOV M, A	; Store the result
4009	CF		RST 1	;stop

Expt.3; 4 BYTE ADDITION

a) Write a program that adds a 4 byte integer stored in consecutive memory locations starting from 4040H beginning with Lower order byte to another 4 byte integer stored in consecutive memory location starting from 4050H beginning with Lower order byte. Store the result in consecutive memory location starting from 4040H. (20)

b) Enter the program on the microprocessor kit. (5)

c) Execute the program. Write the contents of data memory locations before and after execution as well as the contents of the registers used in the program after execution and also the bit contents of all five flags individually. Verify the result. (5)

Addrs	Opcode	Label	Mnemonics	Comment
4000	AF		XRA A	;resets carry flag
4001	06 04		MVI B, 04H	;set counter in reg.B
4003	21 40 40		LXI H, 4040H	; set memory pointer in HL
4006	11 50 40		LXI D, 4050H	; set memory pointer in DE
4009	1A	UP:	LDAX D	;load Acc. with data from address of DE

400A	8E		ADC M	;add it with data from memory pointed by HL
400B	77		MOV M, A	;store the result
400C	13		INX D	;Increment Address in DE
400D	23		INX H	;Increment Address in HL
400E	05		DCR B	;Decrement counter
400F	C2 09 40		JNZ UP	Jump if not zero to UP
4012	CF		RST 1	;stop

Expt. 4 : MULTIPLICATION /DIVISION

a) Write a program that multiplies TWO 1 byte Hex numbers stored in consecutive memory locations starting from 4050H. Store the two byte result in the consecutive memory locations starting from 4052H beginning with Lower order byte. (20)

b) Enter the program on the microprocessor kit. (5)

c) Execute the program. Write the contents of data memory locations before and after execution as well as the contents of the registers used in the program after execution and also the bit contents of all five flags individually. Verify the result. (5)

Addrs	Opcode	Label	Mnemonics	Comment
4000	21 50 40		LXI H, 4050H	;set memory pointer
4003	5E		MOV E, M	;move data from A to E
4004	16 00		MVI D, 00H	;Set counter D to 0
4006	23		INX H	;increment memory address
4007	7E		MOV A, M	;take data in Acc.
4008	06 08		MVI B, 08H	;Set counter B to 8
400A	21 00 00		LXI H, 0000H	;Load HL=00
400D	17	MVLT:	RAL	;rotate left
400E	D2 12 40		JNC ADD1	;jump if no carry to ADD1
4011	19		DAD D	;direct addition
4012	05	ADD 1:	DCR B	;Decrement counter B
4013	CA 1A 40		JZ STORE	;jump if finished to STORE
4016	29		DAD H	;Direct addition
4017	C3 0D 40		JMP MVLT	;jump to repeat
401A	22 52 40	STORE:	SHLD 4052H	;store result at memory.
401D	CF		RST 1	;stop

Input data:4050 09H ;multiplicant
4051 04 ;multiplier

Result : 4052 24H = (09x04=36) in Decimal.

Expt.5 DIVISION

a) Write a program that divides TWO 1 byte Hex numbers where the dividend is stored in 4060H and the divisor is stored in 4061H. Store the quotient and the remainder in the next consecutive memory locations respectively. (20)

b) Enter the program on the microprocessor kit. (5)

c) Execute the program. Write the contents of data memory locations before and after execution as well as the contents of the registers used in the program after execution and also the bit contents of all five flags individually. Verify the result. (5)

Addrs	Opcode	Label	Mnemonics	Comment
4000	21 60 40		LXI H, 4060H	;set memory pointer
4003	7E		MOV A, M	;move data to Acc.
4004	23		INX H	; increment memory address
4005	46		MOV B, M	; move data to reg. B
4006	0E 00		MVI C, 00H	;set counter C=0
4008	90	LOOP:	SUB B	;subtract data of B from A
4009	0C		INR C	;increment counter
400A	B8		CMP B	; compare with B
400B	D2 08 40		JNC LOOP	;jump if not finished to Loop
400E	23		INX H	; increment memory pointer
400F	71		MOV M, C	;Store C in memory
4010	23		INX H	;Increment memory pointer
4011	77		MOV M, A	;Store result in memory
4012	CF		RST 1	;stop

Input Data: 4060 90H Output: 4062 30H (144÷03 =48)in decimal
4061 03H

Expt. 6: BCD ADDITION

a) Write a program that adds the BCD contents of a block of memory. Block length in Hex not exceeding $63_h = 99_{10}$ is stored at 4050H and starting address of block is 4051H. Store the BCD sum result starting from memory location 4060H. (20)

b) Enter the program on the microprocessor kit. (5)

c) Execute the same. Write the contents of data memory locations before and after execution as well as the contents of the registers used in the program after execution and also the bit contents of all five flags individually. Verify the result. (5)

Addrs	Opcode	Label	Mnemonics	Comment
4000	21 50 40		LXI H, 4050H	;set memory pointer
4003	0E 05		MVI C, 05H	;set counter C= 0
4005	AF		XRA A	;reset carry flag to 0
4006	47		MOV B, A	;set B=0
4007	CD 20 40	NEXT:	CALL BCDAD	;call subroutine
400A	23		INX H	; Increment memory pointer
400B	0D		DCR C	;decrement counter C
400C	C2 07 40		JNZ NEXT	; jump if not finished to NEXT
400F	21 60 40		LXI H, 4060H	; set memory pointer

4012	CD 30 40		CALL UNPCK	; call subroutine
4015	78		MOV A, B	;move B to A
4016	CD 30 40		CALL UNPCK	; call subroutine
4019	CF		RST 1	;restart

Subroutine BCDAD:

4020	86	BCDAD:	ADD M	;add data in A to mem. Data
4021	27		DAA	;decimal adjust Acc.
4022	D0		RNC	;return if no carry
4023	57		MOV D, A	;move data from A to D
4024	78		MOV A, B	;move data from B to A
4025	C6 01		ADI 01H	; Add 01
4027	27		DAA	;decimal adjust
4028	47		MOV B, A	; move data from A to B
4029	7A		MOV A, D	; move data from D to A
402A	C9		RET	;return to main program

Subroutine UNPCK:

4030	57	UNPCK:	MOV D, A	; move data from A to D
4031	E6 0F		ANI 0FH	;make logical AND
4033	77		MOV M, A	; move data from A to mem.
4034	2B		DCX H	;decrement memory pointer
4035	7A		MOV A, D	; move data from D to A
4036	E6 F0		ANI F0H	;make logical AND
4038	0F		RRC	;Rotate right
4039	0F		RRC	;Rotate right
403A	0F		RRC	;Rotate right
403B	0F		RRC	;Rotate right
403C	77		MOV M, A	; move data from A to mem.
403D	2B		DCX H	;decrement memory pointer
403E	C9		RET	;return

DATA INPUT		OUTPUT RESULT		
4050	13H		405F	07H
4051	14H		4060	05H
4052	15H			
4053	16H			
4054	17H			

Expt.7:BLOCK TRANSFER

a) A block of data is stored in memory locations from 4050H to 405FH. Write the program to transfer the data in reverse order to memory location starting from 4060H. (20)

b) Enter the program on the microprocessor kit. (5)

c) Execute the same. Write the contents of both data blocks before and after execution as well as the contents of the registers used in the program after execution and also the bit contents of all five flags individually. Verify the results. (5)

Addrs	Opcode	Label	Mnemonics	Comment
4000	06 0F		MVI B, 0FH	;set counter to B
4002	21 5F 40		LXI H, 405FH	;set memory pointer HL
4005	11 60 40		LXI D, 4060H	;set second memory pointer DE
4008	7E	UP:	MOV A, M	;move data from mem. To Acc.
4009	12		STAX D	;store it at memory add. of DE
400A	13		INX D	;increment DE address
400B	2B		DCX H	;decrement HL address
400C	05		DCR B	;decrement counter B
400D	C2 08 40		JNZ UP	;jump if not finished to UP
4010	CF		RST 1	;stop

Expt.8: EXCHANGE OF THE CONTENTS OF THE BLOCK

a) A block of data is stored in memory locations from 4050H to 405FH. Another block of data having the same length is stored in memory locations from 4040H. Write the program to exchange the contents of these two blocks. (20)

b) Enter the program on the microprocessor kit. (5)

c) Execute the same, write the contents of both data blocks before and after execution as well as the contents of the registers used in the program after execution and also the bit contents of all five flags individually. Verify the results. (5)

Addrs	Opcode	Label	Mnemonics	Comment
4000	21 50 40		LXI H, 4050H	;set memory pointer in HL
4003	11 40 40		LXI D, 4040H	;set second memory pointer in DE
4006	06 05		MVI B, 0FH	;set counter B=05
4008	1A	UP:	LDAX D	;load Acc. from mem. of DE
4009	4F		MOV C, A	;move Acc. to C
400A	7E		MOV A, M	;move data of HL mem. to A
400B	12		STAX D	;Store acc. data to mem. of DE
400C	71		MOV M, C	;move C to memory of HL
400D	23		INX H	;increment HL address
400E	13		INX D	;increment DE address
400F	05		DCR B	;decrement counter
4010	C2 08 40		JNZ UP	;Jump if not finished to UP
4013	CF		RST 1	;stop

Expt.9: NIBBLE/ HEX ROTATE

a) Write a program that separates the two nibbles of a number stored in ... and stores the same in memory locations ... and ... The program must also multiply the two nibbles and stores the product in ...... (20)

b) Enter the program on the microprocessor kit. (5)

c) Execute the same. Write the contents of the data memory locations before and after execution as well as the contents of the registers used in the program after execution and also the bit contents of all flags individually. Verify the results. (5)

Addrs	Opcode	Label	Mnemonics	Comment
4000	21 40 40		LXI H, 4040H	;set memory pointer in HL
4003	7E		MOV A, M	;move data of HL mem. to A
4004	5F		MOV E, A	;move Acc. to E
4005	E6 F0		ANI F0H	;make logical AND
4007	0F		RRC	;Rotate right
4008	0F		RRC	;Rotate right
4009	0F		RRC	;Rotate right
400A	0F		RRC	;Rotate right
400B	57		MOV D, A	;move Acc. to D as counter
400C	23		INX H	;increment HL address
400D	77		MOV M, A	;move data of HL mem. to A
400E	7B		MOV A, E	;move E to Acc.
400F	E6 0F		ANI 0FH	; make logical AND
4011	23		INX H	;increment HL address
4012	77		MOV M, A	;move data of HL mem. to A
4013	15		DCR D	;decrement counter
4014	47		MOV B, A	;move Acc. to B
4015	88	MUL:	ADD A	;add with Acc.
4016	15		DCR D	;decrement counter
4017	C2 15 40		JNZ MUL	;Jump if not finished to MUL
401A	23		INX H	;increment HL address
401B	77		MOV M, A	;move Acc. to mem.
401C	CF		RST 1	;stop
DATA INPUT				
4040	94			
OUTPUT RESULT				
4041	09H			;upper nibble BCD
4042	04H			;lower nibble BCD
4043	24H			;result in hex

Expt.10: CODE CONVERSION

a) 2- digit BCD number to Binary conversion.

Addrs	Opcode	Label	Mnemonics	Comment
4000	21 50 40		LXI H, 4050H	;set memory pointer in HL
4003	01 60 40		LXI B, 4060H	;set second memory pointer in BC
4006	7E		MOV A, M	;move mem. to Acc.
4007	C5		PUSH B	;Store BC to stack
4008	D5		PUSH D	;Store DE to stack

4009	47		MOV B, A	;move Acc. to B
400A	E6 0F		ANI 0FH	;make logical AND
400C	4F		MOV C, A	;move Acc. to C
400D	78		MOV A, B	;move B. to Acc.
400E	E6 F0		ANI F0H	;make logical AND
4010	0F		RRC	;Rotate right
4011	0F		RRC	;Rotate right
4012	0F		RRC	;Rotate right
4013	0F		RRC	;Rotate right
4014	57		MOV D, A	;move Acc. to D
4015	AF		XRA A	;reset carry flag
4016	1E 0A		MVI E, 0AH	;set E=0A
4018	83	SUM:	ADD E	;add A+E
4019	15		DCR D	;Decrement D
401A	C2 18 40		JNZ SUM	; Jump if not finished to SUM
401D	89		ADD C	;add A+C
401E	D1		POP D	;restore data of stack to DE
401F	C1		POP B	;restore data of stack to BC
4020	02		STAX B	;store Acc. to B
4021	CF		RST 1	;stop
DATA INPUT			**OUTPUT RESULT**	
4050	99H		4060	63H

b) Binary to BCD conversion:

Addrs	Opcode	Label	Mnemonics	Comment
4000	21 50 40		LXI H, 4050H	;set memory pointer in HL
4003	7E		MOV A, M	;move mem. to Acc
4004	23		INX H	;increment HL address
4005	06 64		MVI B, 64H	;set reg.B=64H
4007	36 FF		MVI M, FFH	; set reg. E=FFH
4009	34	UP:	INR M	;increment mem. data
400A	90		SUB B	;subtract A-B
400B	D2 09 40		JNC UP	;jump to UP
400E	80		ADD B	;Add A+B
400F	23		INX H	;increment HL address
4010	06 0A		MVI B, 0AH	;load B=0AH
4012	36 FF		MVI M, FFH	; load mem.=FFH
4014	34	UP1:	INR M	;increment mem. data
4015	90		SUB B	;subtract A-B
4016	D2 14 40		JNC UP1	;jump to UP1
4019	80		ADD B	;Add A+B
401A	23		INX H	;increment HL address
401B	77		MOV M, A	;move mem. to Acc
401C	CF		RST 1	;stop

Input Data: **OUTPUT RESULT**

4050	90H	4051	01H first place
		4052	04H second place
		4053	04H third place

Explanation: (90H) = $(144)_{10}$ = 01 04 04

c)HEX to ASCII conversion.

Addrs	Opcode	Label	Mnemonics	Comment
4000	21 50 40		LXI H, 4050H	;set memory pointer in HL
4003	11 60 40		LXI D, 4060H	;set second memory pointer in DE
4006	7E		MOV A, M	;move mem. to Acc
4007	47		MOV B, A	;move Acc. to B
4008	0F		RRC	;Rotate right
4009	0F		RRC	;Rotate right
400A	0F		RRC	;Rotate right
400B	0F		RRC	;Rotate right
400C	CD 20 40		CALL ASCII	;call subroutine ASCII
400F	12		STAX D	;store Acc.to DE Address
4010	13		INX D	;increment DE address
4011	78		MOV A, B	;move B to Acc.
4012	CD 20 40		CALL ASCII	;call subroutine ASCII
4015	12		STAX D	;store Acc.to DE Address
4016	CF		RST 1	;stop
Subroutine:ASCII				
4020	E6 0F	ASCII:	ANI 0FH	;logical AND
4022	FE 0A		CPI 0AH	;compare with 0AH
4024	DA 29 40		JC CODE	; jump if carry
4027	C6 07		ADI 07H	;Direct addition with 07H
4029	C6 30	CODE:	ADI 30H	;Direct addition with 30H
402B	C9		RET	;return

DATA INPUT:

4050	38H	; Hex Data Input

OUTPUT RESULT

4060	33H	; ASCII of upper nibble 3
4061	38H	; ASCII of lower nibble 8

d)ASCII to HEX conversion.

Addrs	Opcode	Label	Mnemonics	Comment
4000	3A 50 40		LDA 4050H	;load Acc. with data of mem.
4003	D6 30		SUI 30H	;subtract 30H from Acc
4005	FE 0A		CPI 0AH	;compare with 0AH
4007	DA 0C 40		JC DOWN	;jump if carry to DOWN

400A	D6 07		SUI 07H	;subtract 07H from Acc
400C	47	DOWN:	MOV B, A	;move B to Acc.
400D	3A 51 40		LDA 4051H	;load Acc. with data of mem.
4010	D6 30		SUI 30H	;subtract 30H from Acc
4012	FE 0A		CPI 0AH	;compare with 0AH
4014	DA 19 40		JC DOWN1	;jump if carry to DOWN1
4017	D6 07		SUI 07H	;subtract 07H from Acc
4019	4F	DOWN1:	MOV C, A	;move Acc to C
401A	78		MOV A, B	;move B to Acc.
401B	07		RLC	;Rotate left
401C	07		RLC	;Rotate left
401D	07		RLC	;Rotate left
401E	07		RLC	;Rotate left
401F	81		ADD C	; add A+C
4020	32 52 40		STA 4052H	; store result at next location
4023	CF		RST 1	;stop
DATA INPUT				
4050	38H			; ASCII Data Input
4051	34H			; ASCII Data Input
OUTPUT RESULT				
4052	84H			;HEX Equivalent

Expt.11:To Find the number of odd and even nos. from the block of numbers.

Addrs	Opcode	Label	Mnemonics	Comment
4000	21 50 40		LXI H, 4050H	;set memory pointer in HL
4003	AF		XRA A	;reset carry flag
4004	06 05		MVI B, 05H	;move 05H to B
4006	0E 00		MVI C, 00H	;set odd counter to00H
4008	16 00		MVI D, 00H	;set even counter to00H
400A	7E	AGAIN:	MOV A, M	;move mem. to Acc.
400B	1F		RAR	;rotate right through carry
400C	D2 13 40		JNC DOWN	;jump if no carry to DOWN
400F	0C		INR C	;increment odd counter
4010	C3 14 40		JMP EVEN	;jump to EVEN
4013	14	DOWN:	INR D	; increment even counter
4014	23	EVEN:	INX H	;increment HL address
4015	05		DCR B	Decrement counter B
4016	C2 0A 40		JNZ AGAIN	;jump if not finished to AGAIN
4019	21 60 40		LXI H, 4060H	;set HL for result
401C	71		MOV M, C	;store odd counter no.
401D	23		INX H	;increment HL address
401E	72		MOV M, D	;store even counter no.
401F	CF		RST 1	;stop

DATA INPUT			OUTPUT RESULT	
4050	11H		4060	01H no.of Odd data
4051	12H		4061	04H no.of Even data
4052	14H			
4053	16H			
4054	18H			

Subroutines to display

Standard subroutine of SDK85 development machine (ANSHUMAN made) to display Address field and Data field.

i) **UPDAD (0363H) :** This subroutine displays the contents of the DE register in the Address field. The contents of all other registers are affected.

ii) **UPDDT (036EH):** This subroutine displays the contents of the Accumulator in the data field. The contents of all other registers are affected.

Note: These subroutines are available with all microprocessor Development kits. But the names of subroutines and their memory addresses are different.

Students are advised to use proper memory addresses available with their kits. Refer microprocessor kit Manuals.

Expt.12: To display the output of a program on data field.

Suppose a program of addition of two nos. the result is stored in reg. B and C

Addrs	Opcode	Label	Mnemonics	Comment
4000	06 05		MVI B,05H	;first data
4002	0E 07		MVI C,07H	;second data
4004	79		MOV A,C	;move data in Acc.
4005	80		ADD B	Add A+B
4006	CD 6E 03		CALL UPDDT	;call subroutine to display
4009	76		HLT	

Result is: display on data field and system is halted.

Expt.13: To display the output of a program on Address field.

Suppose a program of addition of two bytes.

Addrs	Opcode	Label	Mnemonics	Comment
4000	21 40 20		LXI H, 2040H	;Load HL with first data
4003	01 30 40		LXI B, 4030H	;Load BC with second data
4006	09		DAD B	;direct two byte addition
4007	54		MOV D,H	; move upper byte of addition in D reg.
4008	5D		MOV E,L	; move lower byte of addition in E reg.
4009	CD 63 03		CALL UPDAD	;call subroutine to display
400C	76		HLT	

Result is: display on Address field and system is halted.

❑❑❑

APPENDIX (A)

The Instruction Set Of 8085/8088 (Code Sheet)

Mnemonic	Hex code	Mnemonic	Hex code	Mnemonic	Hex code
ACI byte	CE	CNZ 16-bit	C4	JPE 16-bit	EA
ADC A	8F	CP 16-bit	F4	JPO 16-bit	E2
ADC B	88	CPE 16-bit	EC	JZ 16-bit	CA
ADC C	89	CPI 8-bit	FE	LDA 16-bit	3A
ADC D	8A	CPO 16-bit	E4	LDAX B	0A
ADC E	8B	CZ 16-bit	CC	LDAX D	1A
ADC H	8C	DAA	27	LHLD 16-bit	2A
ADC L	8D	DAD B	09	LXI B,16-bit	01
ADC M	8E	DAD D	19	LXI D,16-bit	11
ADD A	87	DAD H	29	LXI H,16-bit	21
ADD B	80	DAD SP	39	MOV SP,16-bit	31
ADD C	81	DCR A	3D	MOV A,A	7F
ADD D	82	DCR B	05	MOV A,B	78
ADD E	83	DCR C	0D	MOV A,C	79
ADD H	84	DCR D	15	MOV A,D	7A
ADD L	85	DCR E	1D	MOV A,E	7B
ADD M	86	DCR H	25	MOV A,H	7C
ANA A	A7	DCR L	2D	MOV A,L	7D
ANA B	A0	DCR M	35	MOV A,M	7E
ANA C	A1	DCX B	0B	MOV B,A	47
ANA D	A2	DCX D	1B	MOV B,B	40
ANA E	A3	DCX H	2B	MOV B,C	41
ANA H	A4	DCX SP	3B	MOV B,D	42
ANA L	A5	DI	F3	MOV B,E	43
ANA M	A6	EI	FB	MOV B,H	44
ANI 1byte	E6	HLT	76	MOV B,L	45
CALL 16-bit	CD	IN 1byte	DB	MOV B,M	46
CC 16-bit	DC	INR A	3C	MOV C,A	4F
CM 16-bit	FC	INR B	04	MOV C,B	48
CMA	2F	INR C	0C	MOV C,C	49
CMC	3F	INR D	14	MOV C,D	4A
CMP A	BF	INR E	1C	MOV C,E	4B
CMP B	B8	INR H	24	MOV C,H	4C
		INX B 03	INX D 13		
CMP C	B9	INX SP	33	MOV C,L	4D
CMP D	BA	JC 16-bit	DA	MOV C,M	4E
CMP E	BB	JM 16-bit	FA	MOV D,A	57
CMP H	BC	JMP 16-bit	C3	MOV D,B	50
CMP L	BD	JNC 16-bit	D2	MOV D,C	51
CMP M	BE	JNZ 16-bit	C2	MOV D,D	52

CNC 8-bit	D4	JP 16-bit	F2	MOV D,E	53
Mnemonic	**Hex code**	**Mnemonic**	**Hex code**	**Mnemonic**	**Hex code**
MOV D,H	54	MVI L, 1byte	2E	RST 6	F7
MOV D,L	55	MVI M, 1byte	36	RST 7	FF
MOV D,M	56	NOP	00	RZ	C8
MOV E,A	5F	ORA A	B7	SBB A	9F
MOV E,B	58	ORA B	B0	SBB B	98
MOV E,C	59	ORA C	B1	SBB C	99
MOV E,D	5A	ORA D	B2	SBB D	9A
MOV E,E	5B	ORA E	B3	SBB E	9B
MOV E,H	5C	ORA H	B4	SBB H	9C
MOV E,L	5D	ORA L	B5	SBB L	9D
MOV E,M	5E	ORA M	B6	SBB M	9E
MOV H,A	67	ORI 1-byte	F6	SBI 1-byte	DE
MOV H,B	60	OUT 1-byte	D3	SHLD 16-bit	22
MOV H,C	61	PCHL	E9	SIM	30
MOV H,D	62	POP B	C1	SPHL	F9
MOV H,E	63	POP D	D1	STA 16-bit	32
MOV H,H	64	POP H	E1	STAX B	02
MOV H,L	65	POP PSW	F1	STAX D	12
MOV H,M	66	PUSH B	C5	STC	37
MOV L,A	6F	PUSH D	D5	SUB A	97
MOV L,B	68	PUSH H	E5	SUB B	90
MOV L,C	69	PUSH PSW	F5	SUB C	91
MOV L,D	6A	RAL	17	SUB D	92
MOV L,E	6B	RAR	1F	SUB E	93
MOV L,H	6C	RC	D8	SUB H	94
MOV L,L	6D	RET	C9	SUB L	95
MOV L,M	6E	RIM	20	SUB M	96
MOV M,A	77	RLC	07	SUI 8-bit	D6
MOV M,B	70	RM	F8	XCHG	EB
MOV M,C	71	RNC	D0	XRA A	AF
MOV M,D	72	RNZ	C0	XRA B	A8
MOV M,E	73	RP	F0	XRA C	A9
MOV M,H	74	RPE	E8	XRA D	AA
MOV M,L	75	RPO	E0	XRA E	AB
MVI A, 1byte	3E	RRC	0F	XRA H	AC
MVI B, 1byte	06	RST 0	C7	XRA L	AD
MVI C, 1byte	0E	RST 1	CF	XRA M	AE
MVI D, 1byte	16	RST 2	D7	XRI 8-bit	EE
MVI E, 1byte	1E	RST 3	DF	XTHL	E3
MVI H, 1byte	26	RST 4	E7		

Appendix(B) HTML

Tag Syntax	Description	Attributes
<html>...</html>	Topmost element which contains html document	-
<head>...</head>	Contents within this are not visible to the user	-
<body>...</body>	Contains the main body content which is to be displayed to the user	BGCOLOR, BACKGROUND, TEXT,LINK,ALINK, VLINK
<H1>...</H1>to <H6>...</H6>	To display headings in a web page	Align
<Font>...</Font>	To change the font, font size ,color	Color, face, size
<OL>...</OL>	For creating a numbered or ordered list	Type, start
<UL>...</UL>	For creating an unordered list	Type
<LI>...</LI>	Defines a list item	Value
<DL>...</DL>	Defines a definition list	-
<DT>...</DT>	Defines a definition term	-
<DD>	Gives the definition of the term	-
<B>	Boldfaces the text	-
<I>	Displays the text in italics	-
<U>	Underlines the text	-
<S> or <Strike>	Strikes through the text	-
<TT>	Gives the text a mono-spaced typewriter font	-

<Big>	Text appears in a larger text size	-
<Small>	Text appears in a smaller text size	-
<Sub>	Displays the text as Subscript	-
<sup>	Displays the text as superscript	-
<BR>	Inserts a line break in a web document	-
<HR>	To display a horizontal line	Align ,size ,width
<P>...</P>	To create a paragraph	Align
<pre>...</pre>	Used to display preformatted text	-
<img>	To display an image in the web document	Src , alt, height , width ,usemap , ismap
<map>...</map>	To define a client-side image map	Name
<meta>	Gives information about the web page mainly to search engines	-
<Table>...</table>	To define a table structure within a web page	Border,Height,width, cellspacing ,cellpadding
<TR>...</TR>	Defines a row of the table	Align,valign,bgcolor
<TD>...</TD>	Creates a table cell data	Rowspan ,colspan ,align, valign
<TH>...</TH>	Displays a table cell heading	Rowspan, colspan, align, valign

SPECIMEN QUESTION PAPERS

MARCH-2004(PAPER I)

Time: 3 Hrs. **Marks: 50**

1.A) Select the correct alternative and rewrite the following: **4**

a)_____ is not an Operating system.

i) UNIX, ii) LINUX, iii) MS-DOS, iv) C^{++}

b)Which of the following is not a feature of Object Oriented Programming?

i) Follows bottom-up approach in program design.

ii) Objects may communicate with each other through functions.

iii) Follows top-down approach in program design.

iv) Programs are divided into what are known as objects.

c)The most efficient search algorithm is _____.

i) Binary search, ii) Reverse search, iii) Linear search, iv) Pointer search

d)VB script can be executed in _____ web browser.

i) Netscape Navigator, ii) Internet Explorer, iii) Both, iv) None of these

B) Answer any two of the following: **6**

a) Explain the following HTML tags with one example of each:
i) <PRE>, ii) <SUP>, iii) <MARQUEE>

b) What do you understand by the term 'Searching'? Which are the different types of searching algorithms? Explain the linear searching algorithm.

c) What is meant by GUI? What are the essential components of GUI? Explain any three.

2.A) Answer any two of the following: **6**

a) What are 'records'? Explain how records are represented in memory using arrays?

b) Explain the use of scope resolution operator and memory management operators in C++ with examples.

c)Explain what is meant by Linked List with a suitable example and a properly labelled diagram.

B) Answer any one of the following: **4**

a) What are classes in C++ for file stream operation? Who do you open and close files in C++? Explain any four file modes.

b) Explain the file system related to Information Management with file operations only.

3.A) Answer any two of the following: **6**

a) Explain how the memory address of a variable can be accessed in C++.

b) What are the components of LINUX operating system? Explain any three features of LINUX.

c) Explain the following OOP concepts with an example of each:
i) Inheritance, ii) Polymorphism, iii) Data Abstraction

B) Answer any one of the following: **4**

a)What are binary trees? Draw the binary tree structure for the following expression:

$E = (a + b) / [c * d) - e]$

b) Explain the concept of 'Virtual Memory'. Explain any three terms used in virtual memory.

4.A) Answer any two of the following: **6**

a) What is the difference between a Worm and a Virus? Explain how these can be prevented.

b)What are pointers in C++? Explain the use of pointer variables for function definitions using call by value and call by reference.

c)What is a call is C++? How are member functions defined inside and outside the class. Explain with examples.

B) Answer any one of the following: **4**

a)What is Operator Overloading? Explain with a suitable example. Why is it necessary to overload an operator?

b) What are the different functions performed by Memory Management in operating system? Draw a memory map of a single user computer. Explain types of partitioning in brief.

5.Answer any two of the following: **10**

a) Write a C++ program to replace every space in an inputted string (less than 80 characters) with a hyphen (i.e. -).

b) Write a C++ program to find factorial of a natural number inputted during program execution.

c) Write the HTML code for the following table:

		Year		
		1999	2000	2001
Sales	Units	300	750	1,200
	Income	Rs. 3,000	Rs. 7,500	Rs. 12,000

OR

5. Answer any two of the following: **10**

a) Write a C++ program to display a series of 15 terms of the Fibonacci series.

b) Write a HTML code using VB script for designing a Web Page which greets "Good Morning" if time is from 12:00 AM upto 12:00 PM, else greets "Good Afternoon".

c) Write the output of the following C++ program:

```
#include<iostream.h>
long comb(int n, int k);
int main( )
{
    const m = 5;
```

```
        for(int i = 0; i < m; i ++)
        {
                for (int j = 1; j < m - i; j++)
                        cout << setw(2) << " ";
                for(int j = 0; j < = i; j++)
                        cout << setw(4) << comb(i, j);
        }
    }
    long comb(int n, int k)
    {
        if(n < 0 || k < 0 || K > n) return 0;
        long c = 1;
        for(int i = 1; j < = k; i++, n- -)
                c = c * n/i;
                return c;
}
```

* * *

MARCH-2008

Q. 1 A) Select the correct alternatives and rewrite then sentences: **4**

1. Terminate a Process is the system call available in ________________.

   a) Process b) Memory c) Information d) File

2. Maximum number of nodes of symmetric binary tree with depth of 7 is ________________.

   a) 125 b) 127 c) 128 d) 124

3. following operator cannot be overloaded ________________.

   a) + + b) : : c) - d) *

4. <A> tag has attribute __________ which defines URL of the document to be linked.

   a) SRC b) HREF c) VREF d) REF

B) Answer any two of the following: **6**

1. Explain the following HTML tags with one example of each:

   i) <MARQUEE> ii) <SUB> iii) <BODY>

2. Explain the following data structures with suitable diagram:

   i) Linear Array ii) Linked List iii) Tree

3. Explain in short the function of menu bar and scroll bar components of GUI.

Q. 2 A) Answer any two of the following: **6**

1. What is a Record? Hot it differs from a Linear Array?
2. What is a Class? Explain general from of class declaration.
3. What is Linked List? Show a linked list with suitable example having six nodes with a properly labeled diagram.

B) Answer any one of the following: 6

1. Explain the concept of function overloading with example.
2. With reference to process management explain the following terms:
   i) External Priority ii) Purchase Priority iii) internal Priority iv) Time Slice

Q. 3 A) Answer any two of the following: 6

1. Describe how member functions of class can be defined outside the class definition and inside the class definition.
2. Which are the three major areas in which the operating system divides its services. Give example.
3. Explain the following OOP concepts with an example of each:
   i) Polymorphism ii) Objects iii) Classes

B) Answer any one of the following: 6

1. What is Binary Tree? Draw the binary tree structure for the following expression:
   [(a + b) * c] / [a * (b – c) + a]
2. What is Partitioning? Explain fixed and variable partitioning.

Q. 4 A) Answer any two of the following: 6

1. Discuss Virus Detection, Removal and Prevention Philosophies.
2. What are pointers in C++? Explain the use of pointer variable for function definition using call by value and call by reference.
3. What is the function of each of the following file stream classes?
   i) ifstream ii) ofstream iii) filebuf

B) Answer any one of the following: 6

1. Explain the use of video RAM. Explain Data Bytes and Attribute Bytes.
2. Explain Bubble Sort Algorithm with suitable example.

Q. 5 Answer any two of the following: 10

1. Implement a Circle Class. Each object of this class will represent a circle accepting its radius value a float. Include an area ( ) function which will calculate the area of circle.
2. Write a C++ program to find factorial of a natural number inputted during program execution.

3. Write the exact output of the following HTML code with font specifications in brackets:

```
<HTML>
<TITLE> INTRODUCTION </title>
<BODY>
<HI> <B> PCMBEC <IB> </H/>
<HR>
<U> NOBAL'S SERIES </U>
<H5> WELLKNOWN </H5>
</BODY>
</HTML>
```

OR

Q. 5 Answer any two of the following: **10**

1. Write a C++ program to find greatest common divisor of two numbers. Define a method find to accept the values and calculate GCD of two numbers and print the GCD value.
2. Write a C++ program that will read a line of text and count the number of words in a text.
3. Write the HTML code for the following table:

		Year		
		2000	2001	2002
Sale	Units	500	1000	1500
	Income	Rs. 5,000	Rs. 10,000	Rs. 15,000

□□□

MARCH-2016

Q. 1 A) Select the correct alternatives and rewrite then sentences: **4**

(i) BORDER is attribute used in_____ tag of HTML

a) <PRE> b) <ADDRESS> c) <TABLE> d) <CAPTION>

(ii) In C++ double type data consumes_____ bytes in memory.

a) 2 b) 4 c) 6 d) 8

(iii) _____ is an operating system.

a) VBSCRIPT b) UNIX c) C d) BASIC

(iv) Context switching is a term related to_____ management.

a) Process b) Memory c) Information d) Device

B) Answer any two of the following: **6**

(a) Explain why user is not allowed to directly interact with the hard disc .due to I\O Operation required to read data from disk.

(b) State any 6 characteristics of constructor.

(c) Define i) File ii) Record iii) Key-field

Q. 2 A) Answer any two of the following: **6**

(a). Explain Virus detection, Removal , Prevention.

(b). What is polymorphism? Explain Compile Time and run time polymorphism.

(c). State algorithm for inserting an element in an array.

B) Answer any one of the following: **6**

(a) What is Operator function ? Explain difference between Operator function as Member Function and as a friend function.

(b) Explain local data and global data or variable in C++ using examples.

Q. 3 A)Answer any two of the following: **6**

(a). State characteristics of friend function.

(b). Explain local data and global data or variable in C++ using example.

(c). State features of WINDOWS NT

.B) Answer any one of the following: **4**

(a) Using examples explain how files are opened and closed in C++. State any four file modes:

(b) Explain the process of Multiprogramming in process management with help of an example.

Q. 4 A) Answer any two of the following: **6**

(a). State three characteristics of static data.

(b). Explain the use of memory management operators in C++.

(c). What is record? How it is represented in memory.

B) Answer any one of the following: **4**

(a) What is Virtual Memory? Explain any three terms related to virtual memory.

(b) Explain the following terms related to memory management.

i) Internal Priority ii) External Priority iii) Purchase Priority iv) Time Slice

Q. 5 A) Answer any two of the following: **10**

(a). Write a program in C++ to print first 20 terms of Fibonacci series.

(b). Implement a class temperature . Include a constructor in it which accepts value of temperature from user in degree Celsius. Include two functions in it,

one of which calculates its equivalent temperature in degree Farhrenheit and other function prints the answer:
[Formula : C/5 = F-32/9]

(c) Write an HTML code for Following:

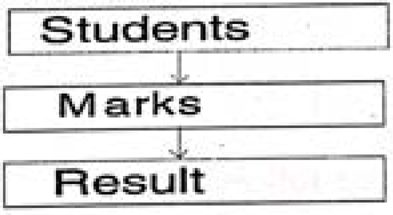
Year	Students		
	Boys	Girls	Total
2004	25	30	55
2005	80	25	105

OR

Q. 5 A) Answer any two of the following: **10**

(a) Implement a class average. Include a constructor in it which will accept value of three variables from user. Include two more functions in it, one function calculates average and other prints it.

(b) Implement the above class Hierarchy of inheritance. Class student accepts roll number of a student, class marks accepts marks of three subjects and class result calculates the total and prints all details.



(c) Write an HTML code for following:

COLLEGE h1 and center

Principal

- Vice Principal
- Professors
- Non-teaching Staff

For more details click here

(The words click here act or hyperlink to next page whose address is "C:\ My Documents\A1.HTML")

❑❑❑

MARCH-2017

Q. 1 A) Select the correct alternatives and rewrite then sentences: 4

(i) The time lost in turning the attention of a processor from one process to another is called as_______ .

a) Circuit switching b) Bandwidth c) Context switching d) Packet switching

(ii) A record is a collection of ______.

a) Files b) Arrays c) Fields d) Maps

(iii) If all the visibility labels are missing then by default members of class are

a) Public b) Protected c) Private d) Void

(iv) ______ tag is used to put a line break in HTML code.

a) <HR> b) <BR> c) <P> d) <LI>

B) Answer any two of the following: 6

(a) What is friend function ? Write any four characteristics of friend function.

(b)Explain in short the three special characteristics of a static data member in a class.

(c) Discuss Virus detection, Removal , Prevention philosophies.

Q. 2 A) Answer any two of the following: 6

(a). Explain with flow chart the following control structure:
i) Sequence logic ii) Selection logic iii) Iteration logic

(b). Explain bubble sort algorithm with suitable example.

(c). What functions are performed by memory management of operating system? State any four memory management system.

B) Answer any one of the following: 6

(a) What is Operator function ? Describe the syntax of operator function. Explain difference between Operator function as Member Function and as a friend function.

(b) Explain the following terms related to process management.

i) Internal Priority ii) External Priority iii) Purchase Priority iv) Time Slice

Q. 3 A)Answer any two of the following: 6

(a). Write difference between linear search and binary search.

(b). Explain different types of inheritance with suitable diagram.

(c). How linked list are represented in memory?

B) Answer any one of the following: 4

(a) Define the following terms with reference to tree:

i) Root ii) leaf iii) sibling iv) Depth

(b) What is computer virus? What are the different methods by which virus can infect other programs.

Q. 4 A) Answer any two of the following: **6**

(a). With syntax diagram explain the structure of HTML webpage.

(b). Explain the concept of function overloading with example.

(c). Explain any three features of Windows-NT operating system.

B) Answer any one of the following: **4**

(a) State the various steps involved in the allocation of partition in case of fixed partition memory management.

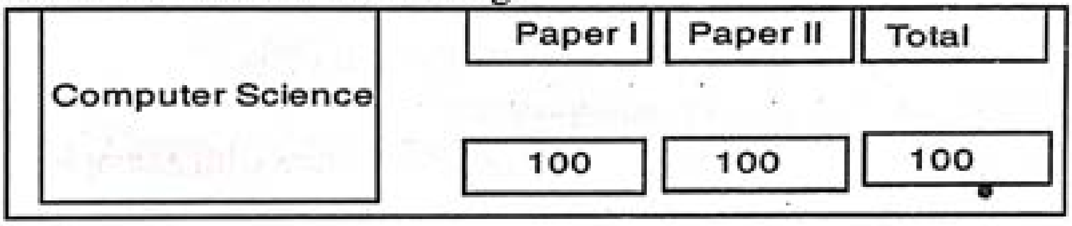
(b) Explain the use of scope resolution operator and memory management operators in C++ with examples.

Q. 5 A) Answer any two of the following: **10**

(a). Write a program in C++ to read a set of 10 numbers from keyboard and find out largest number in the given array.

(b). Write a program in C++ to find factorial of entered number.

(c) Write an HTML code for Following:



Computer Science	Paper I	Paper II	Total
	100	100	100

OR

Q. 5 A) Answer any two of the following: **10**

(a) Implement class GCD which have member function (a/c), which calculates greatest common divisor of two numbers entered during program execution. Print() will print GCD of two numbers.

(b) Write a program in C++ to display a series of 15 term of Fibonacci series.

(c) Write the exact output of the following HTML code with font specification in bracket.

```
<html>
<title>Introduction</title>
 <body>
<h1><b> computer science</b></h1>
 <hr>
 <u> Paper I </u>
 <hr>
 <u> Paper II </u>
 </body>
 </html>
```

□□□

MARCH-2019

1.A) Select the correct option from the following and rewrite sentence : 1

a) Bulleted list in HTML is created by ______ tag.

i) <UL>, ii) <OL>, iii) <B>, iv) <BR>

b) ______ is very useful in situation when data is to be stored and retrieved in reverse order. 1

i) Stack. ii) Queue, iii) Linked List, iv) Tree

c) What will be the value of x after execution of following expression in C++? 1

X = ++ m + n + +; where m = 10 and n = 15.

i) 25, ii) 27, iii) 26, iv) 28

d) ______ is free software. 1

(i) UNIX (ii) WINDOWS

(iii) LINUX (iv) DOX

1.B) Answer any two of the following :

a) List any six features of LINUX Operating System. 3

b) What is a pointer in C++? Give suitable example. 3

c) Write advantages and disadvantages of HTML. 3

2.A) Answer any two of the following :

a) Explain the syntax of C++ program structure with example. 3

b) Explain Binary Search algorithm with a suitable example. 3

c) Write the difference between Worm and Virus. 3

2.B) Answer any one of the following :

a) Explain operator overloading with suitable example. Write ant two characteristics of the operator overloading. 4

b) What is Binary Tree? Draw the Tree diagram for the expression.

$B = (3R/5T)^2 - (R + Q^3)$

3.A) Answer any two of the following :

a) What is a constructor in C++? State any four special characteristics of constructor function. 3

b) Explain the function of the following in Operating System : 3

(i) Virus Detection (ii) Virus Removal (iii) Virus Prevention

c) Write two features of each of data structure : 3

(i) Record (ii) Array (iii) Linked List

3.B) Answer any one of the following :

a) Explain any two types of conversion in C++ with example. 4

b) What are the functions of Memory Management? State any two types of continuous Real Memory Management System. 4

4.A) Answer any two of the following :

a) Explain any six operators used in C++. 3

b) Write a short note on Paging. 3

c) Explain Pointer Array with example. 3

4.B) Answer any one of the following :

a) With reference to process management, explain the terms : 4

(i) External Priority (ii) Purchase Priority

(iii) Internal Priority (iv) Time Slice

b) Explain memory representation of linked list with example. 4

5. Solve any two of the following :

a) Write a C++ program to accept a sentence (maximum 50 characters) and print sentence in reverse. 5

b) Write a function in C++ to accept four integers. Find the smallest integer and print it. 5

c) Write exact output of the following HTML code :

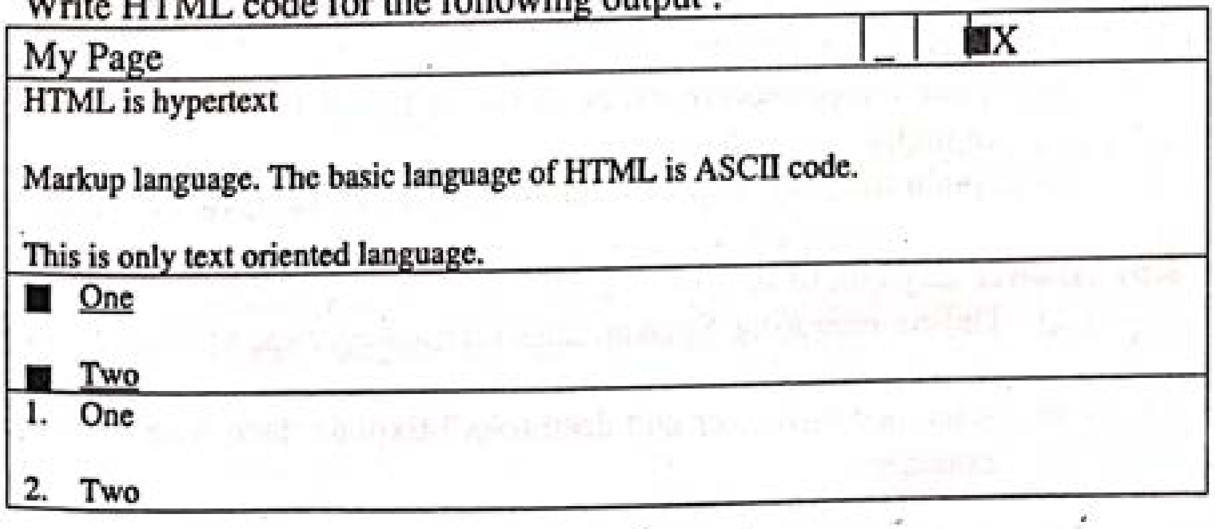
```
<HTML>
    <BODY>
            <OL start = "10">
                    <li> English
                    <li> Second language
            </OL>
            <OL Type = "a">
                    <li> Compulsory
                    <li> Optional
            </OL>
            <UL type = "Square">
                    <Li> Science
                    <Li> Arts
                    <Li> Commerce
            </UL>
            </BODY>
</HTML>
```

5. Solve any two of the following :

a) Write a C++ program to find smallest in an array of 10 flats using pointer. 5

b) Write a class based program in C++ to find area of a Triangle. 5

c) Write HTML code for the following output :



My Page

HTML is hypertext

Markup language. The basic language of HTML is ASCII code.

This is only text oriented language.

- ■ One
- ■ Two

1. One
2. Two

OR

5. **Solve any two of the following :**
   a) Write a C++ function to accept two integers and find its G.C.D. (Greatest Common Divisor). 5
   b) Write a class based C++ program to find the area of a sphere. 5
   c) Write the HTML code for the following output : 5

(1) Computer Science Theory
  (i) Paper 1 - 50 Marks
  (ii) Paper 2 - 50 Marks
(2) Computer Science Practicals
  (i) Paper 1 - 50 Marks
  (ii) Paper 2 - 50 Marks

❑❑❑

MARCH-2020

1.A) Select the correct option from the following and rewrite the sentence : 4

a) Linux is ______ type of software.
i) Public ii) Free iii) Shareware iv) Licence

b) ______ is collection of fields.
i) File ii) Record iii) Array iv) Queue

c) ______ operator cannot be overloaded.
i) ++ ii) + iii) :: iv) >>

d) ______ tag is used to create a row in table.
i) <td> ii) <th> iii) <tr> iv) <tt>

1.B) Answer any two of the following :

a) Explain any three features of windows-98 Operating System. 3
b) Explain Bubble sort algorithm with suitable examples. 3
c) Explain general structure of HTML page. 3

2.A) Answer any two of the following :

a) Explain friend function in C++ with example. 3
b) Explain linked representation of binary tree in memory with suitable example. 3
c) Explain memory map of suitable user operating system. 3

2.B) Answer any one of the following :

a) Define operating System. In which categories, operating system provide its services. 4
b) What is constructor and destructor? Explain each with the help of suitable example. 4

3.A) **Answer any two of the following :**

a) Explain internal and external fragmentation in memory management of operating system. 3

b) List any six data structure operations. 3

c) Define following terms in C++ file handling.
i) Ifstream ii) Ofstream iii) Fstream

3.B) **Answer any one of the following :**

a) What is virus? Write any three infecting methods of virus. 4

b) Define Binary tree. Draw a Tree diagram for following expression. 4
$Y = [(a - b - c) + (a + b - c)]^{3/2}$

4.A) **Answer any two of the following :**

a) Explain time sharing related to process management of operating system. 3

b) Explain how a member function is defined outside class with examples. 3

c) Write C++ declaration for the following. 3
i) Array of 10 integers ii) Pointer to character variable III
iii) Object of the class

4.B) **Answer any one of the following :**

a) Define linked list. Draw and explain labelled diagram of linked list with 5 nodes. 4

b) What is polymorphism? Explain how it is achieved by : 4
i) Compile time ii) Run Time

Q.5) **Answer any two of the following :**

a) Write a C++ program to accept 10 integers in an array and find its sum and average.

b) Write a C++ function to find surface area of a sphere.

c) Write a code in HTML for following table.

Subject		Paper-I	Paper-II
Computer	Theory	50	50
Science	Practical	50	50

OR

Q.5) **Answer any two of the following :**

a) Write a C++ program to accept a sentence of 80 characters and count number of word in a sentence. 5

b) Write a class based C++ program to accept two integers and find its G.C.D. (Greatest Common Factor) 5

c) Write the exact output of the following HTML code with font specification. 5

(Note : here code is given for which output is required along with link details as mentioned)

```
<HTML>
<BODY>
<TABLE border= "3" Cellspacing= "10">
<TR>
<TH colspan = "3" > STREAM </TH>
</TR>
<TR>
<TD> <A href= "Science.html" > SCIENCE </A> </TD>
<TD> <A href= "Commerce.html" > COMMERCE </A> </TD>
<TD> <A href= "Arts.html" > ARTS </A> </TD>
</TR>
</TABLE>
</BODY></HTML>
```

□□□

SPECIMEN QUESTION PAPERS -II

Time: 3 Hrs. **MARCH-2004(PAPER II)** **Marks: 50**

1.A) Select the correct alternative and rewrite the following: **4**

a) In 8085 microprocessor, serial data from external device is received on _____ pin.
   i) SID, ii) SOD, iii) HOLD, iv) READY

b) In 8085 microprocessor, flag register is not affected after the execution of _____ instruction.
   i) INR r, ii) DCR r, iii) ADD r, iv) INX rp

c) _____ is not a characteristic feature of 8051 microcontroller.
   i) 4kbyte of internal RAM, ii) 4 kbyte of internal ROM
   iii) 4 parallel bi-directional I / O part, iv) Full featured serial port

d) A device used for modulation and demodulation process in network is
   i) Hub, ii) Router, iii) Modem, iv) Repeater

B) Answer any two of the following: **6**

a) Explain the organization of ALU with simple block diagram.
   Explain the register and direct addressing modes of 8085 microprocessor with an example of each.

b) List any six major features of 8051 microcontroller.

2.A) Answer any two of the following: **6**

a) Explain the function of the following registers of 8085 microprocessor.
   i) Instruction register, ii) Accumulator, iii) Program counter

b) Explain the following instructions of 8085 microprocessor with suitable example of each:
   i) SHLD addr, ii) ANI data, iii) RRC

c) Explain in short the six important characteristics of transmission media.

B) Answer any one of the following: **4**

a) Explain in brief the four primary functions of the CPU of a microcomputer.

b) Compare any four attributes of 80286 and Pentium microprocessor.

3.A) Answer any two of the following: **6**

a) Explain the sign and parity flags of 8085 microprocessor with suitable example.

b) Describe in brief the function of the following pins of 8085 microprocessor:
   i) REST OUT, ii) HLDA, iii) $IO/\overline{M}$

c) What do you mean by network topology? Explain in brief the two basic categories of topology.

B) Answer any one of the following: **4**

a) In 8085 microprocessor, the flag register content is 3CH. Interpret it's meaning.

b) What is Microcontroller? State three expanded features of 8052 over 8051 microcontroller.

4.A) Answer any two of the following: **6**

a) The accumulator of 8085 microprocessor contains the data 45H and register 'E' contains the data 7BH. What will be the content of accumulator after execution of each of following instructions independently?
i) XRA E, ii) ADI C5H, iii) ORI 5BH

b) Explain the following instructions of 8085 microprocessor with suitable example of each: i) PUSH rp, ii) DAD rp

c) What is a Hub? Explain the active and passive Hubs.

B) Answer any one of the following: **4**

a) Compare any four attributes of UTP and Optical Fibre Cable.

b) What do you mean by 'Protocol'? Write a short note a TCP/IP protocol.

5.Answer any two of the following: **10**

a) Write an assembly language program to divide a hexadecimal number stored in a memory location 8000H by a hexadecimal number stored in memory location 8001H. Store the equation at 8002H and remainder at 8003H.

b) An 8-bit number is stored in memory location C400H. Write an assembly language program to count the 'zero' in the given number. Store the count in memory location C500H.

c) Write an assembly language program to transfer first 10 bytes of memory block starting from 5000H to a new block starting from 5020H.

OR

5.Answer any two of the following: **10**

a) Write an assembly language program to generate the Fibonacci's series for first eight numbers. Store the series in a memory block starting from C100H.
(**Note:** The first eight hex numbers of series are 00,01,02,03,05,08,OD)

b) The two BCD numbers are stored at 3400H and 3401H. Write an assembly language program to add these BCD numbers and store the result in memory locations 3402H and 3403H.

c) Write an assembly language program to count the occurrence of the data 9CH in a memory block starting from 4000H to 400FH. Store the count at memory location 4500H.

MARCH-2008

Q. 1 A) Select the correct alternatives and rewrite then sentences: **4**

1. The 8051 internal ROM is ________________.

   a) Found in the Data memory Space b) Used to store variable program data
   c) 4 KBytes of ROM in the Program Memory Space d) All of the above

2. The instruction ________________ will affect the zero flag without changing the contents of the accumulator.

   a) MVI A, 00 b) SUB A c) XRA A d) CMP A

3. ____________ bus is one way data path from MPU to all devices.
   a) Data b) Address c) Control d) None of these
4. Most widely used and economical cable for network installation is ______
   a) Fiber-optic b) UTP c) STP d) Co-axial

B) Answer any two of the following: 6

1. Draw a neat labeled functional block diagram of 8085 Microprocessor.
2. Write a short note on Evolution of Microprocessor giving one example of each generation.
3. Draw a neat labeled diagram of 8051 memory Register Map.

Q. 2 A) Answer any two of the following: 6

1. Describe the following instruction of 8085 microprocessor:
   i) XCHG ii) RAR iii) ADC r
2. Compare the characteristics of Fiber-optic and Co-axial Cable.
3. Write the features of 8085 Microprocessor.

B) Answer any one of the following: 6

1. What is the function of the following units in 8085 Microprocessor:
   i) Program counter ii) Stack Pointer
   ii) Incrementer / Decrementer iv) General purpose Register
2. Explain the programming model of 32- bit version of X-86 family of Microprocessors.

Q. 3 A) Answer any two of the following: 6

1. Describe in brief the function of the following pins of 8085 Microprocessor:
   i) HLDA ii) READY iii) RST 7.5
2. List any three primary functions of the CPU of the Microcomputer.
3. Explain RING Topology and token Passing.

B) Answer any one of the following: 6

1. Write a short note on Flag Register of 8085 Microprocessor. Explain the significance of flag bits with one example.
2. Explain the Memory register Map of 8051 microcontroller with the help of a neat diagram.

Q. 4 A) Answer any two of the following: 6

1. The accumulator contains the data AB_H. What will be the contents after the execution of the following instructions independently:
   i) XRI B5 H ii) CMA iii) SUB A
2. Explain the following instructions of 8085 Microprocessor with suitable example:

i) DAA ii) LHLD

3. What is meant by Protocol? Explain the concept TCP/IP Protocol.

B) Answer any one of the following: 6

1. Explain the following characteristics of Transmission Media:
   i) Band Width ii) Band usage
   iii) Attenuation iv) Immunity for Electromagnetic Induction
2. Write the function of each of the following devices in short:
   i) Modem ii) Repeater iii) Hub iv) Router

Q. 5 Answer any two of the following: 5

1. Write an assembly language program to copy a block of data having starting address 2000_H to a new destination with starting address 3000_H. Length of the block is stored at $1FFF_H$.
2. A block of data is stored in memory starting from memory location $D001_H$. Write an assembly language program to sort the contents of block in ascending order.
3. Write an assembly language program to exchange the two hexadecimal digits of a number stored at memory location 2500_H. Store the new number at memory location 2501_H.

OR

Q. 5 Answer any two of the following: 5

1. A block of data is stored in memory locations from $C080_H$. Length of block is stored at $C07F_H$. Write an assembly language program that searches for the first occurrence of data byte AB_H in the given block. Store the address of the occurrence in HL register pair. If number is not found, then HL register pair must contain $FFFF_H$.
2. Write an assembly language program to perform the multiplication of two eight bit numbers where multiplicand is stored at memory location 2501_H and 2502_H. Multiplier is stored at 2503_H. Result is to be stored at memory location 2504_H and 2505_H.
3. Write a sub-routine to fill the memory locations 2800_H to $28FF_H$ with the hexadecimal numbers 00_H to FF_H respectively.

MARCH-2016

Q. 1 A) Select the correct alternatives and rewrite then sentences: 4

1. ______________ is a Microcontroller
   a) 8086 b) 8051 c) 8088 d) 80286

2. ________________ instruction does not affect the flags.

   a) RAR b) CMPC c) XRA d) MOV A,B

3. If length of cable is very long then —— is used in between the weakened signal to its original level.

   a) MODEM b) HUB c) REPEATER d) ROUTER

4. ________________Instruction is used for 16-bit addition.

   a) ADD b) ADI c) ADC d) DAD

B) Answer any two of the following: **6**

1. Differentiate between Micro-controller and Microprocessor.
2. Explain the following i) Accumulator ii) Program counter iii) Stack Pointer
3. Write a short note on MODEM

Q. 2 A) Answer any two of the following: **6**

1. Describe in brief the function of the following pins of 8085 microprocessor:
   i) HLDA, ii) SID iii)READY
2. Discuss in brief the members of X-86 family beginning from 80386.
3. Draw memory map of Micro-controller-8051.

B) Answer any one of the following: **6**

1, Draw the labeled internal block diagram of 8085 microprocessor.

2. Explain in brief programming model of X-86 family.

Q. 3 A) Answer any two of the following: **6**

1. Explain any three addressing modes of 8085 with examples.
2. Explain in short: a) Star topology b) Bus topology c) Ring topology
3. Distinguish between LAN and WAN.

B) Answer any one of the following: **4**

1, What is vectored interrupt? State different hardware interrupts with their priorities and branching address.

2. Explain the following features of Pentium processor

a) dual pipelining b) Perfecting c) branch Prediction d) Internal data bus.

Q. 4 A) Answer any two of the following: **6**

1. What is protocol? Explain the concept of TCP/IP protocol.
2. Explain the structure of fiber optic cable.
3. Draw the labeled diagram of X-86 family flag register.

B) Answer any one of the following: **4**

1, Discuss the micro-controllers in 8051 family

2. Write a note on Ethernet.

Q. 5 A) Answer any two of the following: **10**

1. Write an assembly language program to perform the multiplication of two eight bit numbers where multiplicand is stored at memory location 1050_H and 1051_H. . Result is to be stored at memory location 1052_H and 1053_H.
2. Write an assembly language program to transfer a memory block starting from 1050H to1059H a new block starting from 1070H to 1079H.
3. A two byte number stored at C000H and C001H.write an assembly language program to Rotate this number to left side 3 places and store the rotated number in BC register.

OR

Q. 5 A) Answer any two of the following: **10**

1, Write an assembly language program to add 2 decimal numbers stored at 1050H and 1051H store the result at 1052H and 1053H.

2. The accumulator contains the data $B7_H$ and register B contains $A5_H$. What will be the contents after the execution of the following instructions independently:

   i)ADI 05 H ii) CMP B iii) CMA iv) XRAB v) ORAB.

3.Write an assembly language program to increment the contents of alternate memory locations each by two from 1051H to 1060H. □□□

MARCH-2017

Q. 1 A) Select the correct alternatives and rewrite then sentences: **4**

(i) The flag register of 8085 microprocessor contains ______ flags.

a) 8 b) 3 c) 7 d) 5

(ii) ANA , r instruction comes under______ group.

a) Arithmetic b) Logical c) Branch d) Data Transfer

(iii) The maximum physical memory can be addressed by 80286 microprocessor is ______.

a) 640KB b) 1MB c) 16MB d) 4KB

(iv) ______ cable uses light signals to transmit data.

a) Fiber Optic b) coaxial c)UTP d) STP

B) Answer any two of the following: **6**

(a) Explain functions of the following pins of 8085 Microprocessor:

i) Multiplexed address/data bus pin(A_{D0}-A_{D7}) ii)RST 6.5 iii)CLK(OUT)

(b) Write the addressing mode and length in bytes of the following instructions:

i) CPI 10H ii) MOV M,B iii) SHLD C009H

(c) Compare any three characteristics of twisted pair cable with coaxial cable.

Q. 2 A) Answer any two of the following: 6

(a). Define the following terms with suitable diagrams:

i) T State ii) Machine Cycle iii)Instruction Cycle

(b). What is wireless Media? Write any two advantages of wireless media.

(c). The accumulator in 8085 microprocessor contains data 71H register E contains data 39H. What will be the contents of accumulator in Hexadecimal after execution of the following instructions independently?

i) ADD E ii) ORA E iii) RRC

B) Answer any one of the following: 6

(a) What is Microcontroller? State any three advanced features of 8052 microcontroller over 8051 microcontroller.

(b) What is Vectored Interrupt? State all hardware interrupts with their vectored addresses, write their priorities of hardware interrupts.

Q. 3 A) Answer any two of the following: 6

(a). Write any three difference points between memory mapped I\O and I\O Mapped I/O Addressing scheme..

(b). Explain the following instructions of 8085 microprocessor with one example of each: i) PUSH PSW ii) INX rp iii) DAD rp

(c). Write short note on modem.

B) Answer any one of the following: 4

(a) Write any two features of following microcontrollers:

i) 8084 ii) 8052 iii) 8031 iv) 8050

(b) What is Ethernet ? Discuss different types of Ethernet.

Q. 4 A) Answer any two of the following: 6

(a) Compare any three attributes of 80386 and 80486 Microprocessor.

(b) Write any three instructions to make accumulator zero.

(c). What is microprocessor? List its functions.

B) Answer any one of the following: 4

(a) Write a function of following functional units of 8085 Microprocessor:

i) Instruction Decoder ii) General Purpose Register

iii) Data/Address Buffer iv) Status Register.

(b)What is transmission media? Explain in short six characteristics of it.

Q. 5 A) Answer any two of the following: **10**

(a). Write a assembly language program to copy a block of data having starting address 4500H to new location starting from 4600 H. The length of block is stored at memory location 44FF H.

(b). Write an assembly language program to add two 8 bits BCD numbers stored at memory location 4500H and 4501 H. Store the two byte BCD result from memory location 4502 H onwards.

(c) Write an Assembly language program to fill the memory locations 4500H to 4504H with the Hexadecimal numbers 09H to ODH respectively.

OR

Q. 5 A) Answer any two of the following: **10**

(a) Write an assembly language program to exchange the nibbles 8-bit number stored in memory location 4500H. Store the result at memory location 4501H.

(b) A block of data is stored in memory location 4500H. The length of block is stored in memory location 44FFH. Write an assembly language program that searches for the first occurrence of data D9H in given block. Store the address of this occurrence in HL pair. If the number is not found then HL pair should contain 5000 H.

(c) A block of data is stored in memory location 4501H and onwards. The length of block is stored in memory location 4500H. Write an assembly language program to find the sum of block of data. Store the two byte result from memory location 4600H.

❑❑❑

MARCH-2019

1.A) Select the correct alternatives and rewrite the following:

a) _______ bits of flag register of 8085 Microprocessor are unused. 1

i) 1 ii) 2 iii) 3 iv) 4

b) The first byte of an 8085 instruction always contains _______. 1

i) Opcode ii) Data

iii) Address iv) None of these

c) The 8081 Micro-controller has instruction set of _______ instructions. 1

i) 101 ii) 110 iii) 99 iv) 111

d) _______ of the following is an example of wireless media. 1

i) Optic Fibre ii) Microwave

iii) UTP iv) STP

1.B) Solve any two of the following :

a) Explain the functions of following pins of 8085 Microprocessor : 3

i) $IO/\overline{M}$ ii) $\overline{RD}$ iii) $\overline{INTR}$

b) Differentiate between LAN and WAN (Any 3 points) 3

c) State any six features of 8051 micro-controller. 3

2.A) Solve any two of the following :

a) Define following terms : 3
i) T-State ii) Machine Cycle iii) Instruction Cycle

b) Considering following points, explain the given instruction : 3
Instruction : INX rp
i) Group of Instruction ii) Addressing Mode
iii) Number of Bytes iv) Flag Affected v) Explain with example

c) Explain Micro-controller and state any two of its advantages over Micro-processor. 3

2.B) Solve any one of the following :

a) Draw a neat and labeled block diagram of Micro-computer. 4

b) Explain STAR Topology with diagram also give two advantages and disadvantages. 4

3.A) Solve any two of the following :

a) Write the functions of following blocks of 8085 Micro-processor : 3
i) Accumulator ii) Instruction Register iii) Decoder

b) Give any two instructions of following addressing modes : 3
i) Immediate ii) Register Indirect iii) Register

c) Explain TCP/IP protocol in detail. 3

3.B) Solve any one of the following :

a) Explain following network devices with diagram : 4
i) Router ii) Repeater

b) Explain memory register map of 8051 Micro-controller with diagram. 4

4.A) Solve any two of the following :

a) What are Hardware Interrupts? List them according to priority. Also state call location of these interrupts. 3

b) Explain ALU of 8085 Micro-processor with suitable diagram. 3

c) Enlist six characteristics of Transmission Media. 3

4.B) Solve any one of the following :

a) The accumulator contains AA H and register C contains 55 H. What will be the contents of accumulator if following instructions are executed independently?4
i) CMP C ii) ANA C iii) ORA C iv) SUB C

b) Compare any four features of 80486 and Pentium Processors. 4

Q. 5) Solve any two of the following :

a) Write an Assembly Language Program to find absolute difference of two hex numbers stored in memory locations 5000H and 5001H. Store the result in 5002H. 5

b) Write an Assembly Language Program to find largest number in a block of memory starting from 7000H. The length of the block is stored at 6FFF H. Store the result at the end of the block. 5

c) Study the following program and answer the questions given below. 5

Label	Mnemonics / Operand
	MVI C,, 08 H
	LXI H, 6000 H
	MOV A, M
BACK	RRC
	DCRC
	INZ BACK
	INX H
	MOV M, A
	HLT

(i) Write the purpose of the program.

(ii) Write comments for the instructions used in the program.

(iii) If the input data at memory location 6000 H is FF H, then write the result along with corresponding memory location.

OR

Q. 5) Solve any two of the following :

a) A block of data is stored in memory locations starting from 3001 H. The length of the block is at 3000 H. Write an Assembly Language Program that searches for the first occurrence of data AO H in given block. Store the address of this occurrence in HL pair. If the number is not found then HL pair should contain 0000 H. 5

b) Write an Assembly Language Program to find sum of ten hex numbers stored in consecutive memory locations from 4000 H. Store the two byte result at the end of the block beginning with lower byte. 5

c) Study the following program and answer the questions given below : 5

Label	Mnemonics / Operand	
	LXI H, COOO H	
	MOV C, M	
	INX H	
BACK	MOV A, M	
	XRA A	
	MOV M, A	
	DCR C	
	JNZ BACK	
	HLT	

(i) Write the purpose of the program.

(ii) Write comments for the instructions used in the program.

(iii) If the input data at memory location C000 H is 05 H, then write the result along with corresponding memory location.

d) Study the following program and answer the questions given below :

```
STC
CMC
LXI B, 1234H
MOV A, B
RAR
MOV H, A
MOV A, C
RAR
MOV L, A
HLT
```

(i) Write the purpose of the program.

(ii) Write the contents of various registers used.

Write comments of various instructions used in the program. ❑❑❑

MARCH-2020

1.A) Select the correct option of the following and rewrite the sentences :

a) _______ is non-maskable interrupt in 8085. 1

i) RST 5.5 ii) RST 6.5
iii) RST 7.5 iv) TRAP

b) The length of instruction MVI reg. data is ______.

i) 1 byte ii) 2 Byte iii) 3 Byte iv) 4 Byte

c) The 8051 Micro-controller can address _______ program memory. 1

i) 8 k byte ii) 16 k byte iii) 32 k Byte iv) 64 k Byte

d) ______ cable is insensetive of EMI.

i) Co-axial ii) STP
iii) UTP iv) Fibre Optic

1.B) Answer any two of the following :

a) Write a note on evolution of Micro-processor. 3

b) Explain any three addressing modes of 8085 Micro-processor with 1 example. 3

c) Write a HUB? Explain Active and Passive HUB. 3

2.A) Answer any two of the following :

a) What is multiplexed BUS in 8085? Give its advantages. 3

b) Explain following instruction of 8085 Micro-processor. 3

i) CMA ii)RRC iii)STC

c) Define Topology. Explain Physical and Logical Topology. 3

2.B) Answer any one of the following :

a) Define following registers of Micro-processor 8085. 4

i) Accumulator ii)STACK Pointer Program Counter iv)Instruction Register

b) Explain following terms related to Pentium Processor : 4

i) Dual Pipeline ii) Branch Prediction iii) On chip cacheiv) 64 bit data BUS

3.A) Answer any two of the following :

a) Explain any three features of 8085 Micro-processor. 3

b) Explain the function of following pins of 8085 Micro-processor : 3
i) X_1, X_2 ii)CLKOUT iii)RD

c) List any six features of 8085 Micro-controller. 3

3.B) Answer any one of the following :

a) Explain memory map of 8051 Micro-controller. 4

b) Explain Contention and Polling Access Methods. 4

4.A) Answer any two of the following :

a) Flag register contain data C5H interpret its meaning. 3

b) The accumulator contains data 58H and register B contains data 07H. What will be the content of Accumulator after execution of following instruction independently : 3
i) ADD B ii) ORA B iii) ANA B

c) Explain Co-axial Cable in detail. 3

4.B) Answer any one of the following :

a) What is Interrupt? List Hardware Interrupts according to Priority. Explain maskable and non-maskable in interrupts. 4

b) Explain the following characteristics of Transmission media : 4
i) Installation Difficulties ii) EMI
iii) Band Width iv) Attenuation.

Q.5) Answer any two of the following :

a) A block of data is stored from memory location D001H. Length of block is stored at D000H. Write a program to find occurrence of data 02H in given block. Store the number of occurrence at Memory Location D100H. 5

b) A block of data is stored from memory location D001H to D005H. Copy the contents of block to another block starting from 2501H. 5

c) Write a program to subtract 3 Byte integer in register EHL from another 3 Byte integer in BCD. The result should be placed in BCD register keeping the integers in EH undisturbed. 5

OR

Q.5) Answer any two of the following :

a) A block of data is stored from memory location 3330H. Length of block is stored at 2FFFH. Write a program to find 2's compliment of each data in a block and store the result from memory location 4100H. 5

b) A block of data is stored from memory location C001H and length is stored in C000H. Write a program to find the sum of series and store the sum in CO50H and CO51H. 5

c) Write a program that divides two 1 byte hex number where the dividend is stored in 4060H and divisor in 406H stored the quotient and remainder in next two consecutive memory location respectively. 5

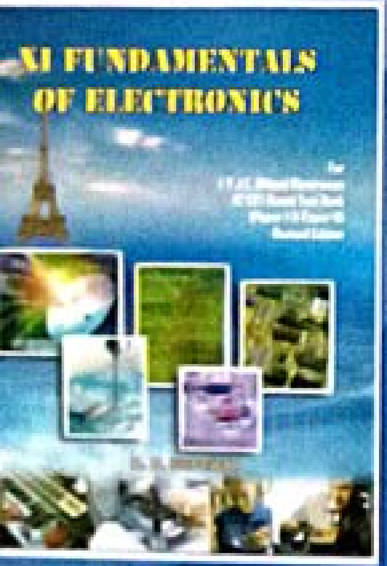
□□□

MAYURESHWAR PRAKASHAN PUNE

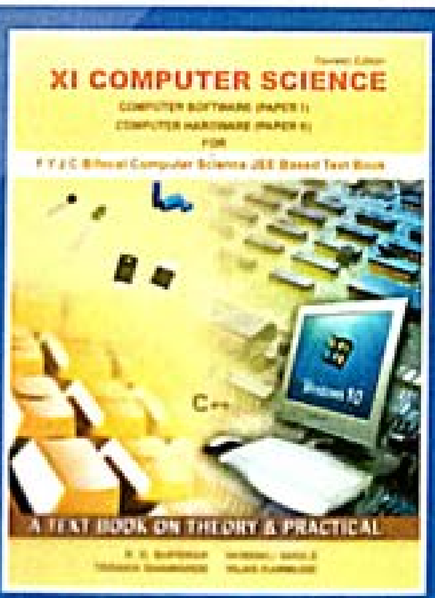
Benefits of Bifocal Computer Science

- Exemption for Second language and optional Subject at XI and XII std.
- Preperation of Engineering subjects at Junior college level
- Easy for XII Physics
- Preperation of I I T / JEE Examination
- Eligible for any Engineering branch (B.E.)
- Eligible for Architecture degree course (B.Arch.)
- Eligible for Pharmacy (B. Pharm.)
- Eligible for Bachelor of Science (B.Sc.) and (BCS)
- Eligible for D. Ed. Course
- Direct Admission to Second year three years Diploma course
- Eligible for B.Tech (Agri)

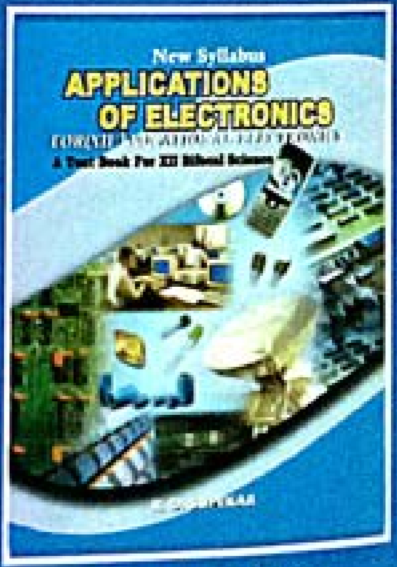
OUR OUTSTANDING BOOKS


XI FUNDAMENTALS OF ELECTRONICS

XI ELECTRONICS
(Paper I & II)


XI COMPUTER SCIENCE
A TEXT BOOK ON THEORY & PRACTICAL

XI Computer Science
(Paper I & II)


New Syllabus
APPLICATIONS OF ELECTRONICS

XII ELECTRONICS
(Paper I & II)

SOLE DISTRIBUTOR

MAYURESHWAR PRAKASHAN, PUNE

" Rajarshi " S.No. 23/1, Anandvihar Colony, Sinhgad Road, Vitthalwadi, Pune 51.
Mobile: 94220 18066

Price Rs.275/-